



## ANSYS Fluent Advanced Add-On Modules



ANSYS, Inc.  
Southpointe  
2600 ANSYS Drive  
Canonsburg, PA 15317  
[ansysinfo@ansys.com](mailto:ansysinfo@ansys.com)  
<http://www.ansys.com>  
(T) 724-746-3304  
(F) 724-514-9494

Release 17.0  
January 2016

ANSYS, Inc. is  
certified to ISO  
9001:2008.

---

## **Copyright and Trademark Information**

© 2015 SAS IP, Inc. All rights reserved. Unauthorized use, distribution or duplication is prohibited.

ANSYS, ANSYS Workbench, Ansoft, AUTODYN, EKM, Engineering Knowledge Manager, CFX, FLUENT, HFSS, AIM and any and all ANSYS, Inc. brand, product, service and feature names, logos and slogans are registered trademarks or trademarks of ANSYS, Inc. or its subsidiaries in the United States or other countries. ICEM CFD is a trademark used by ANSYS, Inc. under license. CFX is a trademark of Sony Corporation in Japan. All other brand, product, service and feature names or trademarks are the property of their respective owners.

## **Disclaimer Notice**

THIS ANSYS SOFTWARE PRODUCT AND PROGRAM DOCUMENTATION INCLUDE TRADE SECRETS AND ARE CONFIDENTIAL AND PROPRIETARY PRODUCTS OF ANSYS, INC., ITS SUBSIDIARIES, OR LICENSORS. The software products and documentation are furnished by ANSYS, Inc., its subsidiaries, or affiliates under a software license agreement that contains provisions concerning non-disclosure, copying, length and nature of use, compliance with exporting laws, warranties, disclaimers, limitations of liability, and remedies, and other provisions. The software products and documentation may be used, disclosed, transferred, or copied only in accordance with the terms and conditions of that software license agreement.

ANSYS, Inc. is certified to ISO 9001:2008.

## **U.S. Government Rights**

For U.S. Government users, except as specifically granted by the ANSYS, Inc. software license agreement, the use, duplication, or disclosure by the United States Government is subject to restrictions stated in the ANSYS, Inc. software license agreement and FAR 12.212 (for non-DOD licenses).

## **Third-Party Software**

See the [legal information](#) in the product help files for the complete Legal Notice for ANSYS proprietary software and third-party software. If you are unable to access the Legal Notice, Contact ANSYS, Inc.

Published in the U.S.A.

---

# Table of Contents

Using This Manual .....	xiii
1.The Contents of the Fluent Manuals .....	xiii
2.Typographical Conventions .....	xiv
3.Mathematical Conventions .....	xvii
4.Technical Support .....	xviii
<b>I. ANSYS Fluent Adjoint Solver .....</b>	<b>1</b>
Using This Manual .....	iii
1.The Contents of This Manual .....	iii
<b>1. Introduction to the Adjoint Solver .....</b>	<b>5</b>
1.1.Overview .....	5
1.1.1.General Observables .....	6
1.1.2.General Operations .....	10
1.2.Discrete Versus Continuous Adjoint Solver .....	11
1.3.Discrete Adjoint Solver Overview .....	12
1.4.Adjoint Solver Stabilization .....	15
1.5.Solution-Based Adaption .....	16
1.6.Using The Data To Improve A Design .....	16
1.6.1.Smoothing and Mesh Morphing .....	17
<b>2. Using the Adjoint Solver Module .....</b>	<b>19</b>
2.1.Installing the Adjoint Solver Module .....	19
2.2>Loading the Adjoint Solver Module .....	20
2.3.Model Considerations for Using Adjoint Solver .....	20
2.3.1.Basic Assumptions and Consistency Checks .....	20
2.3.2.User-Defined Sources .....	21
2.4.Defining Observables .....	22
2.4.1.Creating New Observables .....	23
2.4.2.Editing Observable Definitions .....	25
2.4.3.Selecting an Observable for Sensitivity Calculation .....	28
2.5.Solving the Adjoint .....	28
2.5.1.Using the Adjoint Solution Methods Dialog Box .....	29
2.5.2.Using the Adjoint Solution Controls Dialog Box .....	30
2.5.2.1.Stabilized Scheme Settings .....	32
2.5.2.1.1.Modal Stabilization Scheme .....	32
2.5.2.1.2.Spatial Stabilization Scheme .....	35
2.5.2.1.3.Dissipation Scheme .....	37
2.5.3.Working with Adjoint Residual Monitors .....	38
2.5.4.Running the Adjoint Calculation .....	38
2.6.Postprocessing of Adjoint Solutions .....	39
2.6.1.Field Data .....	39
2.6.2.Scalar Data .....	44
2.7.Modifying the Geometry Using the Design Tool .....	44
2.7.1.Defining the Region for the Design Change .....	46
2.7.2.Defining Region Conditions .....	47
2.7.3.Exporting Sensitivity Data .....	47
2.7.4.Defining Observable Objectives .....	48
2.7.5.Defining Conditions for the Deformation .....	49
2.7.6.Shape Modification .....	53
2.7.7.Design Tool Numerics .....	56
2.8.Using the Adjoint Solver Module's Text User Interface .....	58
<b>3.Tutorial: Using the Adjoint Solver – 2D Laminar Flow Past a Cylinder .....</b>	<b>63</b>

<b>II. ANSYS Fluent Battery Module .....</b>	65
Using This Manual .....	lxvii
1. The Contents of This Manual .....	lxvii
<b>1. Introduction .....</b>	69
1.1. Overview .....	69
1.2. General Procedure .....	70
1.3. Installing the Battery Module .....	70
<b>2. Single-Potential Empirical Battery Model .....</b>	71
2.1. Single-Potential Empirical Battery Model Theory .....	71
2.1.1. Introduction .....	71
2.1.2. Computation of the Electric Potential and Current Density .....	71
2.1.3. Thermal and Electrical Coupling .....	73
2.2. Using the Single-Potential Empirical Battery Model .....	73
2.2.1. Geometry Definition for the Single-Potential Empirical Battery Model .....	73
2.2.2. Loading the Single-Potential Empirical Battery Module .....	74
2.2.3. Getting Started With the Single-Potential Empirical Battery Model .....	74
2.2.3.1. Specifying Single-Potential Empirical Battery Model Parameters .....	75
2.2.3.2. Specifying Separator Parameters .....	78
2.2.3.3. Specifying Electric Field Parameters .....	78
2.2.4. Solution Controls for the Single-Potential Empirical Battery Model .....	79
2.2.5. Postprocessing the Single-Potential Empirical Battery Model .....	80
2.2.6. User-Accessible Functions .....	81
2.2.6.1. Compiling the Customized Battery Source Code .....	82
2.2.6.1.1. Compiling the Customized Source Code Under Linux .....	82
2.2.6.1.2. Compiling the Customized Source Code Under Windows .....	83
2.2.6.7. Using the Single-Potential Empirical Battery Model Text User Interface .....	84
<b>3. Dual-Potential MSMD Battery Model .....</b>	85
3.1. Dual-Potential MSMD Battery Model Theory .....	85
3.1.1. MSMD approach .....	85
3.1.2. NTGK Model .....	86
3.1.3. ECM Model .....	87
3.1.4. Newman's P2D Model .....	88
3.1.5. Coupling Between CFD and Submodels .....	91
3.1.6. Battery Pack Simulation .....	92
3.1.7. Reduced Order Solution Method (ROM) .....	94
3.1.8. External and Internal Electric Short-Circuit Treatment .....	94
3.1.9. Thermal Abuse Model .....	95
3.2. Using the Dual-Potential MSMD Battery Model .....	97
3.2.1. Limitations .....	97
3.2.2. Geometry Definition for the Dual-Potential MSMD Battery Model .....	97
3.2.3. Loading the Dual-Potential MSMD Battery Module .....	98
3.2.4. Setting up the Dual-Potential MSMD Battery Model .....	98
3.2.4.1. Specifying Battery Model Options .....	100
3.2.4.2. Specifying Battery Model Parameters .....	106
3.2.4.2.1. Inputs for the NTGK Empirical Model .....	107
3.2.4.2.2. Inputs for the Equivalent Circuit Model .....	108
3.2.4.2.3. Inputs for the Newman's P2D Model .....	110
3.2.4.2.4. Input for the User-Defined E-Model .....	113
3.2.4.3. Specifying Conductive Zones .....	114
3.2.4.4. Specifying Electric Contacts .....	115
3.2.4.5. Specifying Advanced Option .....	116
3.2.4.6. Specifying External and Internal Short-Circuit Resistances .....	118

3.2.5. Initializing the Battery Model .....	118
3.2.6. Modifying Material Properties .....	119
3.2.6.1. Specifying UDS Diffusivity for the Active Material .....	119
3.2.6.2. Specifying UDS Diffusivity for the Passive Material .....	119
3.2.6.3. Defining Different Materials for Positive and Negative Electrodes .....	119
3.2.7. Solution Controls for the Dual-Potential MSMD Battery Model .....	120
3.2.8. Postprocessing the Dual-Potential MSMD Battery Model .....	120
3.2.9. User-Accessible Functions .....	121
3.2.9.1. Compiling the Customized Battery Source Code .....	122
3.2.9.1.1. Compiling the Customized Source Code Under Linux .....	123
3.2.9.1.2. Compiling the Customized Source Code Under Windows .....	124
3.2.10. Using the Dual-Potential MSMD Battery Model Text User Interface .....	124
<b>4. Tutorial: Simulating a Single Battery Cell Using the MSMD Battery Model .....</b>	127
<b>5. Tutorial: Simulating a 1P3S Battery Pack Using the MSMD Battery Model .....</b>	129
Bibliography .....	131
<b>III. ANSYS Fluent Continuous Fiber Module .....</b>	133
Using This Manual .....	CXXXV
1. The Contents of This Manual .....	CXXXV
<b>1. Introduction .....</b>	137
<b>2. Continuous Fiber Model Theory .....</b>	139
2.1. Introduction .....	139
2.2. Governing Equations of Fiber Flow .....	139
2.3. Discretization of the Fiber Equations .....	142
2.3.1. Under-Relaxation .....	142
2.4. Numerical Solution Algorithm of Fiber Equations .....	143
2.5. Residuals of Fiber Equations .....	143
2.6. Coupling Between Fibers and the Surrounding Fluid .....	144
2.6.1. Momentum Exchange .....	144
2.6.2. Mass Exchange .....	145
2.6.3. Heat Exchange .....	145
2.6.4. Radiation Exchange .....	146
2.6.5. Under-Relaxation of the Fiber Exchange Terms .....	146
2.7. Fiber Grid Generation .....	146
2.8. Correlations for Momentum, Heat and Mass Transfer .....	147
2.8.1. Drag Coefficient .....	148
2.8.2. Heat Transfer Coefficient .....	149
2.8.3. Mass Transfer Coefficient .....	150
2.9. Fiber Properties .....	150
2.9.1. Fiber Viscosity .....	150
2.9.1.1. Melt Spinning .....	151
2.9.1.2. Dry Spinning .....	151
2.9.2. Vapor-Liquid Equilibrium .....	151
2.9.3. Latent Heat of Vaporization .....	152
2.9.4. Emissivity .....	152
2.10. Solution Strategies .....	152
<b>3. Using the Continuous Fiber Module .....</b>	155
3.1. Installing the Continuous Fiber Module .....	155
3.2. Loading the Continuous Fiber Module .....	155
3.3. Getting Started With the Continuous Fiber Module .....	157
3.3.1. User-Defined Memory and the Adjust Function Setup .....	157
3.3.2. Source Term UDF Setup .....	158
3.4. Fiber Models and Options .....	158

3.4.1. Choosing a Fiber Model .....	159
3.4.2. Including Interaction With Surrounding Flow .....	160
3.4.3. Including Lateral Drag on Surrounding Flow .....	160
3.4.4. Including Fiber Radiation Interaction .....	160
3.4.5. Viscous Heating of Fibers .....	160
3.4.6. Drag, Heat and Mass Transfer Correlations .....	160
3.5. Fiber Material Properties .....	161
3.5.1. The Concept of Fiber Materials .....	161
3.5.2. Description of Fiber Properties .....	161
3.6. Defining Fibers .....	163
3.6.1. Overview .....	164
3.6.2. Fiber Injection Types .....	164
3.6.3. Working with Fiber Injections .....	165
3.6.3.1. Creating Fiber Injections .....	166
3.6.3.2. Modifying Fiber Injections .....	166
3.6.3.3. Copying Fiber Injections .....	166
3.6.3.4. Deleting Fiber Injections .....	166
3.6.3.5. Initializing Fiber Injections .....	166
3.6.3.6. Computing Fiber Injections .....	166
3.6.3.7. Print Fiber Injections .....	166
3.6.3.8. Read Data of Fiber Injections .....	167
3.6.3.9. Write Data of Fiber Injections .....	167
3.6.3.10. Write Binary Data of Fiber Injections .....	167
3.6.3.11. List Fiber Injections .....	167
3.6.4. Defining Fiber Injection Properties .....	168
3.6.5. Point Properties Specific to Single Fiber Injections .....	172
3.6.6. Point Properties Specific to Line Fiber Injections .....	172
3.6.7. Point Properties Specific to Matrix Fiber Injections .....	172
3.6.8. Define Fiber Grids .....	173
3.6.8.1. Equidistant Fiber Grids .....	173
3.6.8.2. One-Sided Fiber Grids .....	173
3.6.8.3. Two-Sided Fiber Grids .....	174
3.6.8.4. Three-Sided Fiber Grids .....	174
3.7. User-Defined Functions (UDFs) for the Continuous Fiber Model .....	175
3.7.1. UDF Setup .....	176
3.7.1.1. Linux Systems .....	176
3.7.1.2. Windows Systems .....	176
3.7.2. Customizing the fiber_fluent_interface.c File for Your Fiber Model Application .....	176
3.7.2.1. Example: Heat Transfer Coefficient UDF .....	177
3.7.2.2. Example: Fiber Specific Heat Capacity UDF .....	178
3.7.3. Compile Fiber Model UDFs .....	178
3.7.3.1. Linux Systems .....	179
3.7.3.2. NT/Windows Systems .....	179
3.7.4. Hook UDFs to the Continuous Fiber Model .....	180
3.8. Fiber Model Solution Controls .....	181
3.9. Postprocessing for the Continuous Fibers .....	183
3.9.1. Display of Fiber Locations and Grid Points .....	183
3.9.2. Exchange Terms of Fibers .....	185
3.9.3. Analyzing Fiber Variables .....	186
3.9.3.1. XY Plots .....	186
3.9.3.2. Fiber Display .....	187
3.9.4. Running the Fiber Module in Parallel .....	189

Bibliography .....	191
<b>IV. ANSYS Fluent Macroscopic Particle Module .....</b>	<b>193</b>
Using This Manual .....	CXCV
1.The Contents of This Manual .....	CXCV
<b>1. Introduction .....</b>	<b>197</b>
<b>2. Macroscopic Particle Model Theory .....</b>	<b>199</b>
2.1. Momentum Transfer to Fluid Flow .....	199
2.2. Fluid Forces and Torques on Particle .....	200
2.3. Particle/Particle and Particle/Wall Collisions .....	201
2.4. Field Forces .....	202
2.5. Particle Deposition and Buildup .....	203
<b>3. Using the Macroscopic Particle Model .....</b>	<b>205</b>
3.1. Overview and Limitations .....	205
3.2. Loading the MPM add-on Module .....	205
3.3. Setting up MPM Model Simulations .....	206
3.4. Modeling Macroscopic Particles .....	207
3.4.1. Specifying Particle Tracking Parameters .....	208
3.4.2. Specifying the Drag Law .....	209
3.4.3. Defining Parameters for Particle-Particle and Particle-Wall Collisions .....	211
3.4.4. Specifying Deposition Parameters .....	212
3.4.5. Specifying Injection Parameters .....	213
3.4.5.1. Defining MPM Injection Properties .....	214
3.4.5.2. Inputs for point Injections .....	216
3.4.5.3. Inputs for plane Injections .....	217
3.4.5.4. Inputs for packing Injections .....	219
3.4.5.5. Inputs for from-file Injections .....	220
3.4.6. Defining Field Forces .....	220
3.4.7. Initializing the MPM model .....	221
Bibliography .....	225
<b>V. ANSYS Fluent Fuel Cell Modules .....</b>	<b>227</b>
Using This Manual .....	CCXXIX
1.The Contents of This Manual .....	CCXXIX
<b>1. PEMFC Model Theory .....</b>	<b>231</b>
1.1. Introduction .....	231
1.2. Electrochemistry Modeling .....	232
1.2.1. The Cathode Particle Model .....	235
1.3. Current and Mass Conservation .....	236
1.4. Water Transport and Mass Transfer in PEMFC .....	236
1.4.1. The Dissolved Phase Model .....	237
1.4.2. The Liquid Phase Model .....	238
1.4.2.1. Liquid Water Transport Equation in the Porous Electrode and the Membrane .....	238
1.4.2.2. Liquid Water Transport Equation in Gas Channels .....	239
1.5. Heat Source .....	240
1.6. Properties .....	240
1.7. Transient Simulations .....	242
1.8. Leakage Current (Cross-Over Current) .....	242
<b>2. Using the PEMFC Model .....</b>	<b>245</b>
2.1. Overview and Limitations .....	245
2.2. Geometry Definition for the PEMFC Model .....	245
2.3. Installing the PEMFC Model .....	246
2.4. Loading the PEMFC Module .....	246
2.5. Setting Up the PEMFC Module .....	246

2.6. Modeling PEM Fuel Cells .....	247
2.6.1. Specifying Model Options .....	249
2.6.2. Specifying Model Parameters .....	251
2.6.3. Specifying Anode Properties .....	255
2.6.3.1. Specifying Current Collector Properties for the Anode .....	255
2.6.3.2. Specifying Flow Channel Properties for the Anode .....	256
2.6.3.3. Specifying Porous Electrode Properties for the Anode .....	257
2.6.3.4. Specifying Catalyst Layer Properties for the Anode .....	258
2.6.3.5. Specifying Micro Porous Layer (Optional) Properties for the Anode .....	260
2.6.3.6. Specifying Cell Zone Conditions for the Anode .....	261
2.6.4. Specifying Electrolyte/Membrane Properties .....	261
2.6.4.1. Specifying Cell Zone Conditions for the Membrane .....	262
2.6.5. Specifying Cathode Properties .....	262
2.6.5.1. Specifying Current Collector Properties for the Cathode .....	262
2.6.5.2. Specifying Flow Channel Properties for the Cathode .....	262
2.6.5.3. Specifying Porous Electrode Properties for the Cathode .....	262
2.6.5.4. Specifying Catalyst Layer Properties for the Cathode .....	263
2.6.5.5. Specifying Micro Porous Layer (Optional) Properties for the Cathode .....	264
2.6.5.6. Specifying Cell Zone Conditions for the Cathode .....	265
2.6.6. Setting Advanced Properties .....	265
2.6.6.1. Setting Contact Resistivities for the PEMFC Model .....	265
2.6.6.2. Setting Coolant Channel Properties for the PEMFC Model (Optional) .....	266
2.6.6.3. Managing Stacks for the PEMFC Model .....	267
2.6.7. Reporting on the Solution .....	268
2.7. PEMFC Model Boundary Conditions .....	269
2.8. Solution Guidelines for the PEMFC Model .....	270
2.9. Postprocessing the PEMFC Model .....	271
2.10. User-Accessible Functions .....	272
2.10.1. Compiling the Customized PEMFC Source Code .....	275
2.10.1.1. Compiling the Customized Source Code Under Linux .....	276
2.10.1.2. Compiling the Customized Source Code under Windows .....	277
2.11. Using the PEMFC Text User Interface .....	277
2.11.1. IV-Curve Calculations Using the Text Interface .....	279
<b>3. Fuel Cell and Electrolysis Model Theory .....</b>	<b>281</b>
3.1. Introduction .....	281
3.1.1. Introduction to PEMFC .....	282
3.1.2. Introduction to SOFC .....	283
3.1.3. Introduction to Electrolysis .....	283
3.2. Electrochemistry Modeling .....	284
3.3. Current and Mass Conservation .....	286
3.4. Heat Source .....	287
3.5. Liquid Water Formation, Transport, and its Effects (PEMFC Only) .....	287
3.6. Properties .....	288
3.7. Transient Simulations .....	290
3.8. Leakage Current (Cross-Over Current) .....	290
<b>4. Using the Fuel Cell and Electrolysis Model .....</b>	<b>291</b>
4.1. Overview and Limitations .....	291
4.2. Geometry Definition for the Fuel Cell and Electrolysis Model .....	291
4.3. Installing the Fuel Cell and Electrolysis Model .....	292
4.4. Loading the Fuel Cell and Electrolysis Module .....	292
4.5. Setting Up the Fuel Cell and Electrolysis Module .....	292
4.6. Modeling Fuel Cells and Electrolysis .....	293

4.6.1. Specifying Model Options .....	295
4.6.2. Specifying Model Parameters .....	297
4.6.3. Specifying Anode Properties .....	298
4.6.3.1. Specifying Current Collector Properties for the Anode .....	299
4.6.3.2. Specifying Flow Channel Properties for the Anode .....	300
4.6.3.3. Specifying Porous Electrode Properties for the Anode .....	301
4.6.3.4. Specifying Catalyst Layer Properties for the Anode .....	302
4.6.3.5. Specifying Cell Zone Conditions for the Anode .....	303
4.6.4. Specifying Electrolyte/Membrane Properties .....	303
4.6.4.1. Specifying Cell Zone Conditions for the Membrane .....	304
4.6.5. Specifying Cathode Properties .....	304
4.6.5.1. Specifying Current Collector Properties for the Cathode .....	304
4.6.5.2. Specifying Flow Channel Properties for the Cathode .....	305
4.6.5.3. Specifying Porous Electrode Properties for the Cathode .....	306
4.6.5.4. Specifying Catalyst Layer Properties for the Cathode .....	307
4.6.5.5. Specifying Cell Zone Conditions for the Cathode .....	308
4.6.6. Setting Advanced Properties .....	308
4.6.6.1. Setting Contact Resistivities for the Fuel Cell and Electrolysis Model .....	308
4.6.6.2. Setting Coolant Channel Properties for the Fuel Cell and Electrolysis Model .....	309
4.6.6.3. Managing Stacks for the Fuel Cell and Electrolysis Model .....	310
4.6.7. Reporting on the Solution .....	312
4.7. Modeling Current Collectors .....	313
4.8. Fuel Cell and Electrolysis Model Boundary Conditions .....	314
4.9. Solution Guidelines for the Fuel Cell and Electrolysis Model .....	315
4.10. Postprocessing the Fuel Cell and Electrolysis Model .....	316
4.11. User-Accessible Functions .....	317
4.11.1. Compiling the Customized Fuel Cell and Electrolysis Source Code .....	320
4.11.1.1. Compiling the Customized Source Code Under Linux .....	320
4.11.1.2. Compiling the Customized Source Code Under Windows .....	321
4.12. Using the Fuel Cell and Electrolysis Text User Interface .....	322
4.12.1. IV-Curve Calculations Using the Text Interface .....	324
<b>5. SOFC Fuel Cell With Unresolved Electrolyte Model Theory .....</b>	<b>327</b>
5.1. Introduction .....	327
5.2. The SOFC With Unresolved Electrolyte Modeling Strategy .....	328
5.3. Modeling Fluid Flow, Heat Transfer, and Mass Transfer .....	329
5.4. Modeling Current Transport and the Potential Field .....	329
5.4.1. Cell Potential .....	330
5.4.2. Activation Overpotential .....	331
5.4.3. Treatment of the Energy Equation at the Electrolyte Interface .....	332
5.4.4. Treatment of the Energy Equation in the Conducting Regions .....	334
5.5. Modeling Reactions .....	334
5.5.1. Modeling Electrochemical Reactions .....	334
5.5.2. Modeling CO Electrochemistry .....	335
<b>6. Using the Solid Oxide Fuel Cell With Unresolved Electrolyte Model .....</b>	<b>337</b>
6.1. Limitation on Modeling Solid Oxide Fuel Cells .....	337
6.2. Installing the Solid Oxide Fuel Cell With Unresolved Electrolyte Model .....	337
6.3. Loading the Solid Oxide Fuel Cell With Unresolved Electrolyte Module .....	337
6.4. Solid Oxide Fuel Cell With Unresolved Electrolyte Module Set Up Procedure .....	338
6.5. Setting the Parameters for the SOFC With Unresolved Electrolyte Model .....	345
6.6. Setting Up the Electrochemistry Parameters .....	347
6.7. Setting Up the Electrode-Electrolyte Interfaces .....	349
6.8. Setting Up the Electric Field Model Parameters .....	350

6.9. User-Accessible Functions for the Solid Oxide Fuel Cell With Unresolved Electrolyte Model .....	351
6.9.1. Compiling the Customized Solid Oxide Fuel Cell With Unresolved Electrolyte Source Code .....	352
6.9.1.1. Compiling the Customized Source Code Under Linux .....	352
6.9.1.2. Compiling the Customized Source Code Under Windows .....	353
6.10. Using the Solid Oxide Fuel Cell With Unresolved Electrolyte Text User Interface .....	353
Bibliography .....	355
<b>VI. ANSYS Fluent Magnetohydrodynamics (MHD) Module .....</b>	357
Using This Manual .....	ccclix
1. The Contents of This Manual .....	ccclix
<b>1. Introduction .....</b>	361
<b>2. Magnetohydrodynamic Model Theory .....</b>	363
2.1. Introduction .....	363
2.2. Magnetic Induction Method .....	364
2.2.1. Case 1: Externally Imposed Magnetic Field Generated in Non-conducting Media .....	364
2.2.2. Case 2: Externally Imposed Magnetic Field Generated in Conducting Media .....	365
2.3. Electric Potential Method .....	365
<b>3. Implementation .....</b>	367
3.1. Solving Magnetic Induction and Electric Potential Equations .....	367
3.2. Calculation of MHD Variables .....	368
3.3. MHD Interaction with Fluid Flows .....	368
3.4. MHD Interaction with Discrete Phase Model .....	368
3.5. General User-Defined Functions .....	368
<b>4. Using the ANSYS Fluent MHD Module .....</b>	369
4.1. MHD Module Installation .....	369
4.2. Loading the MHD Module .....	369
4.3. MHD Model Setup .....	370
4.3.1. Enabling the MHD Model .....	371
4.3.2. Selecting an MHD Method .....	372
4.3.3. Applying an External Magnetic Field .....	372
4.3.4. Setting Up Boundary Conditions .....	376
4.3.5. Solution Controls .....	378
4.4. MHD Solution and Postprocessing .....	379
4.4.1. MHD Model Initialization .....	379
4.4.2. Iteration .....	380
4.4.3. Postprocessing .....	380
4.5. Limitations .....	381
A. Guidelines For Using the ANSYS Fluent MHD Model .....	383
A.1. Installing the MHD Module .....	383
A.2. An Overview of Using the MHD Module .....	383
B. Definitions of the Magnetic Field .....	387
C. External Magnetic Field Data Format .....	389
D. MHD Module Text Commands .....	391
Bibliography .....	393
<b>VII. ANSYS Fluent Population Balance Module .....</b>	395
Using This Manual .....	cccxcvii
1. The Contents of This Manual .....	cccxcvii
<b>1. Introduction .....</b>	399
1.1. The Discrete Method .....	399
1.2. The Inhomogeneous Discrete Method .....	399
1.3. The Standard Method of Moments .....	401
1.4. The Quadrature Method of Moments .....	402

<b>2. Population Balance Model Theory .....</b>	403
2.1.The Particle State Vector .....	403
2.2.The Population Balance Equation (PBE) .....	403
2.2.1.Particle Growth and Dissolution .....	404
2.2.2.Particle Birth and Death Due to Breakage and Aggregation .....	404
2.2.2.1.Breakage .....	405
2.2.2.2.Luo and Lehr Breakage Kernels .....	406
2.2.2.3.Ghadiri Breakage Kernels .....	407
2.2.2.4.Laakkonen Breakage Kernels .....	407
2.2.2.5.Parabolic PDF .....	408
2.2.2.6.Generalized PDF .....	408
2.2.2.7.Aggregation .....	411
2.2.2.8.Luo Aggregation Kernel .....	412
2.2.2.9.Free Molecular Aggregation Kernel .....	412
2.2.2.10.Turbulent Aggregation Kernel .....	412
2.2.3.Particle Birth by Nucleation .....	413
2.3.Solution Methods .....	414
2.3.1.The Discrete Method and the Inhomogeneous Discrete Method .....	414
2.3.1.1.Numerical Method .....	414
2.3.1.2.Breakage Formulations for the Discrete Method .....	416
2.3.2.The Standard Method of Moments (SMM) .....	417
2.3.2.1.Numerical Method .....	417
2.3.3.The Quadrature Method of Moments (QMOM) .....	418
2.3.3.1.Numerical Method .....	418
2.3.4.The Direct Quadrature Method of Moments (DQMOM) .....	419
2.3.4.1.Numerical Method .....	419
2.4.Population Balance Statistics .....	421
2.4.1.Reconstructing the Particle Size Distribution from Moments .....	421
2.4.2.The Log-Normal Distribution .....	422
<b>3. Using the ANSYS Fluent Population Balance Model .....</b>	423
3.1.Population Balance Module Installation .....	423
3.2>Loading the Population Balance Module .....	423
3.3.Population Balance Model Setup .....	424
3.3.1.Enabling the Population Balance Model .....	424
3.3.1.1.Generated DQMOM Values .....	432
3.3.2.Defining Population Balance Boundary Conditions .....	435
3.3.2.1.Initializing Bin Fractions With a Log-Normal Distribution .....	436
3.3.3.Specifying Population Balance Solution Controls .....	437
3.3.4.Coupling With Fluid Dynamics .....	438
3.3.5.Specifying Interphase Mass Transfer Due to Nucleation and Growth .....	439
<b>4. Postprocessing for the Population Balance Model .....</b>	443
4.1.Population Balance Solution Variables .....	443
4.2.Reporting Derived Population Balance Variables .....	443
4.2.1.Computing Moments .....	444
4.2.2.Displaying a Number Density Function .....	444
<b>5. UDFs for Population Balance Modeling .....</b>	447
5.1.Population Balance Variables .....	447
5.2.Population Balance DEFINE Macros .....	447
5.2.1.DEFINE_PB_BREAK_UP_RATE_FREQ .....	448
5.2.1.1.Usage .....	448
5.2.1.2.Example .....	448
5.2.2.DEFINE_PB_BREAK_UP_RATE_PDF .....	449

5.2.2.1. Usage .....	449
5.2.2.2. Example .....	449
5.2.3. DEFINE_PB_COALESCENCE_RATE .....	450
5.2.3.1. Usage .....	450
5.2.3.2. Example .....	450
5.2.4. DEFINE_PB_NUCLEATION_RATE .....	451
5.2.4.1. Usage .....	451
5.2.4.2. Example .....	451
5.2.5. DEFINE_PB_GROWTH_RATE .....	452
5.2.5.1. Usage .....	452
5.2.5.2. Example .....	452
5.3. Hooking a Population Balance UDF to ANSYS Fluent .....	453
A. DEFINE_HET_RXN_RATE Macro .....	455
A.1. Description .....	455
A.2. Usage .....	455
A.3. Example .....	456
A.4. Hooking a Heterogeneous Reaction Rate UDF to ANSYS Fluent .....	457
Bibliography .....	459

---

# Using This Manual

---

This preface is divided into the following sections:

1. The Contents of the Fluent Manuals
2. Typographical Conventions
3. Mathematical Conventions
4. Technical Support

## 1. The Contents of the Fluent Manuals

The manuals listed below form the Fluent product documentation set. They include descriptions of the procedures, commands, and theoretical details needed to use Fluent products.

- [Fluent Getting Started Guide](#) contains general information about getting started with using Fluent and provides details about starting, running, and exiting the program.
- [Fluent Migration Manual](#) contains information about transitioning from the previous release of Fluent, including details about new features, output changes, and text command list changes.
- [Fluent User's Guide](#) contains detailed information about running a simulation using the solution mode of Fluent, including information about the user interface, reading and writing files, defining boundary conditions, setting up physical models, calculating a solution, and analyzing your results.
- [ANSYS Fluent Meshing Migration Manual](#) contains information about transitioning from the previous release of Fluent Meshing, including descriptions of new features and text command list changes.
- [ANSYS Fluent Meshing User's Guide](#) contains detailed information about creating 3D meshes using the meshing mode of Fluent.

Related video help can be found on the [ANSYS How To Videos](#) page.

- [Fluent in Workbench User's Guide](#) contains information about getting started with and using Fluent within the Workbench environment.
- [Fluent Theory Guide](#) contains reference information for how the physical models are implemented in Fluent.
- [Fluent Customization Manual](#) contains information about writing and using user-defined functions (UDFs).
- [Fluent Tutorial Guide](#) contains a number of examples of various flow problems with detailed instructions, commentary, and postprocessing of results.

The latest updates of the ANSYS Fluent tutorials are available on the ANSYS Customer Portal. To access tutorials and their input files on the ANSYS Customer Portal, go to <http://support.ansys.com/training>.

- The latest updates of the ANSYS Fluent Meshing tutorials are available on the ANSYS Customer Portal. To access tutorials and their input files on the ANSYS Customer Portal, go to <http://support.ansys.com/training>.
- [Fluent Text Command List](#) contains a brief description of each of the commands in Fluent's solution mode text interface.

- [ANSYS Fluent Meshing Text Command List](#) contains a brief description of each of the commands in Fluent's meshing mode text interface.
- [ANSYS Fluent Advanced Add-On Modules](#) contains information about the usage of the different advanced Fluent add-on modules, which are applicable for specific modeling needs.
  - [Part I: ANSYS Fluent Adjoint Solver](#) contains information about the background and usage of Fluent's Adjoint Solver Module that allows you to obtain detailed sensitivity data for the performance of a fluid system.
  - [Part II: ANSYS Fluent Battery Module](#) contains information about the background and usage of Fluent's Battery Module that allows you to analyze the behavior of electric batteries.
  - [Part III: ANSYS Fluent Continuous Fiber Module](#) contains information about the background and usage of Fluent's Continuous Fiber Module that allows you to analyze the behavior of fiber flow, fiber properties, and coupling between fibers and the surrounding fluid due to the strong interaction that exists between the fibers and the surrounding gas.
  - [Part V: ANSYS Fluent Fuel Cell Modules](#) contains information about the background and the usage of two separate add-on fuel cell models for Fluent that allow you to model polymer electrolyte membrane fuel cells (PEMFC), solid oxide fuel cells (SOFC), and electrolysis with Fluent.
  - [Part VI: ANSYS Fluent Magnetohydrodynamics \(MHD\) Module](#) contains information about the background and usage of Fluent's Magnetohydrodynamics (MHD) Module that allows you to analyze the behavior of electrically conducting fluid flow under the influence of constant (DC) or oscillating (AC) electromagnetic fields.
  - [Part VII: ANSYS Fluent Population Balance Module](#) contains information about the background and usage of Fluent's Population Balance Module that allows you to analyze multiphase flows involving size distributions where particle population (as well as momentum, mass, and energy) require a balance equation.
- [Fluent as a Server User's Guide](#) contains information about the usage of Fluent as a Server which allows you to connect to a Fluent session and issue commands from a remote client application.
- [Running ANSYS Fluent Using a Load Manager](#) contains information about using third-party load managers with ANSYS Fluent.
  - [Part I: Running ANSYS Fluent Under LSF](#) contains information about using Fluent with Platform Computing's LSF software, a distributed computing resource management tool.
  - [Part II: Running ANSYS Fluent Under PBS Professional](#) contains information about using Fluent with Altair PBS Professional, an open workload management tool for local and distributed environments.
  - [Part III: Running ANSYS Fluent Under SGE](#) contains information about using Fluent with Univa Grid Engine (formerly Sun Grid Engine) software, a distributed computing resource management tool.

## 2. Typographical Conventions

Several typographical conventions are used in this manual's text to help you find commands in the user interface.

- Different type styles are used to indicate graphical user interface items and text interface items. For example:

**Iso-Surface** dialog box

surface/iso-surface text command

- The text interface type style is also used when illustrating exactly what appears on the screen to distinguish it from the narrative text. In this context, user inputs are typically shown in boldface. For example,

```
solve/initialize/set-fmg-initialization

Customize your FMG initialization:
  set the number of multigrid levels [5]

  set FMG parameters on levels ..

  residual reduction on level 1 is: [0.001]
  number of cycles on level 1 is: [10] 100

  residual reduction on level 2 is: [0.001]
  number of cycles on level 2 is: [50] 100
```

- Mini flow charts are used to guide you through the ribbon or the tree, leading you to a specific option, dialog box, or task page. The following tables list the meaning of each symbol in the mini flow charts.

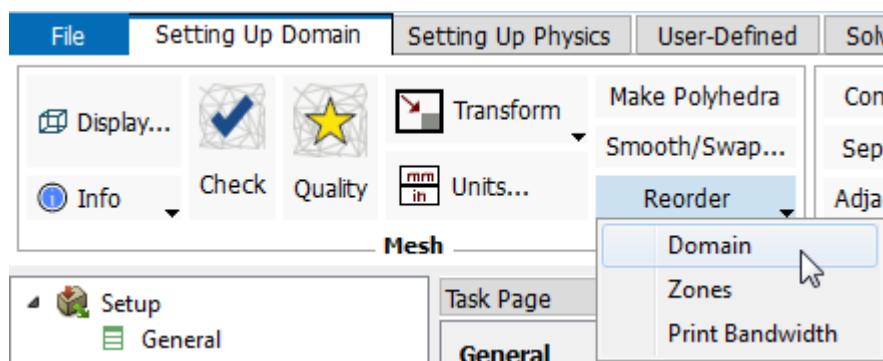
**Table 1: Mini Flow Chart Symbol Descriptions**

Symbol	Indicated Action
	Look at the ribbon
	Look at the tree
	Left-click to open task page
	Select from task page
	Right-click the preceding item

For example,

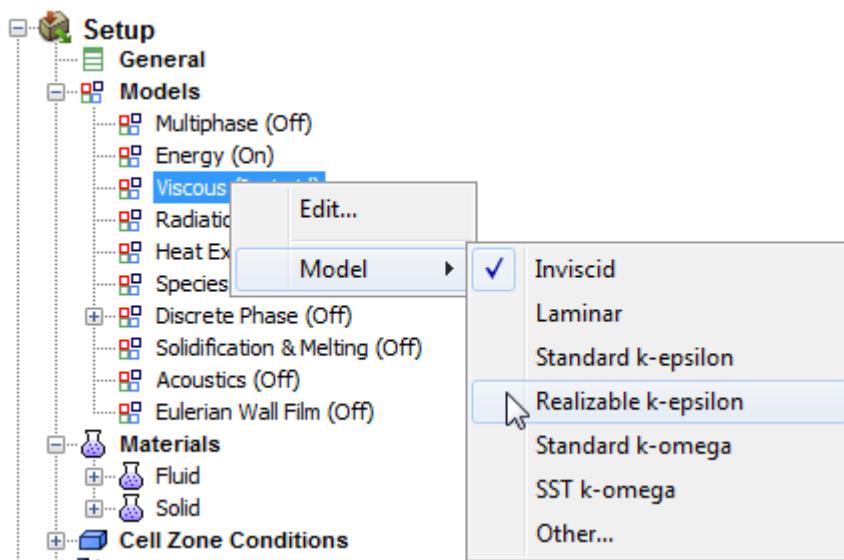
 **Setting Up Domain** → **Mesh** → **Reorder** → **Domain**

Indicates selecting the **Setting Up Domain** ribbon tab, clicking **Reorder** (in the **Mesh** group box) and selecting **Domain**, as indicated in the figure below:



 **Setup** → **Models** → **Viscous**  **Model** → **Realizable k-epsilon**

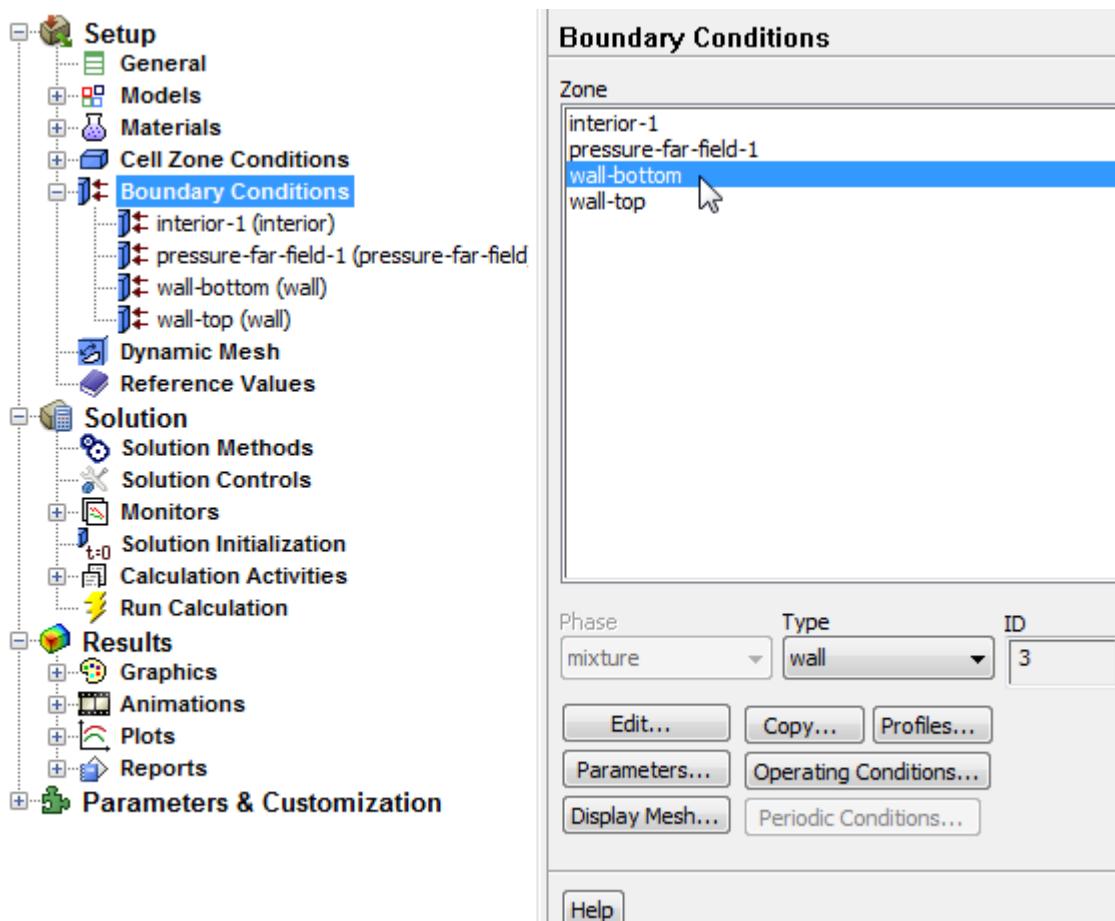
Indicates expanding the **Setup** and **Models** branches, right-clicking **Viscous**, and selecting **Realizable k-epsilon** from the **Model** sub-menu, as shown in the following figure:



And

**Setup** → **Boundary Conditions** → **wall-bottom**

Indicates opening the task page as shown below:



In this manual, mini flow charts usually accompany a description of a dialog box or command, or a screen illustration showing how to use the dialog box or command. They show you how to quickly access a command or dialog box without having to search the surrounding material.

- In-text references to **File** ribbon tab selections can be indicated using a "/". For example **File/Write/Case...** indicates clicking the **File** ribbon tab and selecting **Case...** from the **Write** submenu (which opens the **Select File** dialog box).

### 3. Mathematical Conventions

- Where possible, vector quantities are displayed with a raised arrow (e.g.,  $\vec{a}$ ,  $\vec{A}$ ). Boldfaced characters are reserved for vectors and matrices as they apply to linear algebra (e.g., the identity matrix,  $\mathbf{I}$ ).
- The operator  $\nabla$ , referred to as grad, nabla, or del, represents the partial derivative of a quantity with respect to all directions in the chosen coordinate system. In Cartesian coordinates,  $\nabla$  is defined to be

$$\frac{\partial}{\partial x} \vec{i} + \frac{\partial}{\partial y} \vec{j} + \frac{\partial}{\partial z} \vec{k} \quad (1)$$

$\nabla$  appears in several ways:

- The gradient of a scalar quantity is the vector whose components are the partial derivatives; for example,

$$\nabla p = \frac{\partial p}{\partial x} \vec{i} + \frac{\partial p}{\partial y} \vec{j} + \frac{\partial p}{\partial z} \vec{k} \quad (2)$$

- The gradient of a vector quantity is a second-order tensor; for example, in Cartesian coordinates,

$$\nabla(\vec{v}) = \left( \frac{\partial}{\partial x} \vec{i} + \frac{\partial}{\partial y} \vec{j} + \frac{\partial}{\partial z} \vec{k} \right) (v_x \vec{i} + v_y \vec{j} + v_z \vec{k}) \quad (3)$$

This tensor is usually written as

$$\begin{pmatrix} \frac{\partial v_x}{\partial x} & \frac{\partial v_x}{\partial y} & \frac{\partial v_x}{\partial z} \\ \frac{\partial v_y}{\partial x} & \frac{\partial v_y}{\partial y} & \frac{\partial v_y}{\partial z} \\ \frac{\partial v_z}{\partial x} & \frac{\partial v_z}{\partial y} & \frac{\partial v_z}{\partial z} \end{pmatrix} \quad (4)$$

- The divergence of a vector quantity, which is the inner product between  $\nabla$  and a vector; for example,

$$\nabla \cdot \vec{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z} \quad (5)$$

- The operator  $\nabla \cdot \nabla$ , which is usually written as  $\nabla^2$  and is known as the Laplacian; for example,

$$\nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} \quad (6)$$

$\nabla^2 T$  is different from the expression  $(\nabla T)^2$ , which is defined as

$$(\nabla T)^2 = \left( \frac{\partial T}{\partial x} \right)^2 + \left( \frac{\partial T}{\partial y} \right)^2 + \left( \frac{\partial T}{\partial z} \right)^2 \quad (7)$$

- An exception to the use of  $\nabla$  is found in the discussion of Reynolds stresses in [Turbulence in the Fluent Theory Guide](#), where convention dictates the use of Cartesian tensor notation. In this chapter, you will also find that some velocity vector components are written as  $u$ ,  $v$ , and  $w$  instead of the conventional  $v$  with directional subscripts.

## 4. Technical Support

If you encounter difficulties while using ANSYS Fluent, please first refer to the section(s) of the manual containing information on the commands you are trying to use or the type of problem you are trying to solve. The product documentation is available from the online help, or from the ANSYS Customer Portal. To access documentation files on the ANSYS Customer Portal, go to <http://support.ansys.com/documentation>.

If you encounter an error, please write down the exact error message that appeared and note as much information as you can about what you were doing in ANSYS Fluent.

Technical Support for ANSYS, Inc. products is provided either by ANSYS, Inc. directly or by one of our certified ANSYS Support Providers. Please check with the ANSYS Support Coordinator (ASC) at your company to determine who provides support for your company, or go to [www.ansys.com](http://www.ansys.com) and select **Contacts > Contacts and Locations**.

If your support is provided by ANSYS, Inc. directly, Technical Support can be accessed quickly and efficiently from the ANSYS Customer Portal, which is available from the ANSYS Website ([www.ansys.com](http://www.ansys.com)) under **Support > Customer Portal**. The direct URL is: [support.ansys.com](http://support.ansys.com).

One of the many useful features of the Customer Portal is the Knowledge Resources Search, which can be found on the Home page of the Customer Portal. To use this feature, enter relevant text (error message, etc.) in the Knowledge Resources Search box and click the magnifying glass icon. These Knowledge Resources provide solutions and guidance on how to resolve installation and licensing issues quickly.

### NORTH AMERICA

#### All ANSYS Products except Esterel, Apache and Reaction Design products

**Web:** Go to the ANSYS Customer Portal (<http://support.ansys.com>) and select the appropriate option.

**Toll-Free Telephone:** 1.800.711.7199 (Please have your Customer or Contact ID ready.)

Support for University customers is provided only through the ANSYS Customer Portal.

### GERMANY

#### ANSYS Mechanical Products

**Telephone:** +49 (0) 8092 7005-55 (CADFEM)

**Email:** [support@cadfem.de](mailto:support@cadfem.de)

#### All ANSYS Products

**Web:** Go to the ANSYS Customer Portal (<http://support.ansys.com>) and select the appropriate option.

**National Toll-Free Telephone:** (Please have your Customer or Contact ID ready.)

German language: 0800 181 8499

English language: 0800 181 1565

Austria: 0800 297 835

Switzerland: 0800 564 318

**International Telephone:** (Please have your Customer or Contact ID ready.)

German language: +49 6151 152 9981

English language: +49 6151 152 9982

**Email:** support-germany@ansys.com

## UNITED KINGDOM

### All ANSYS Products

**Web:** Go to the ANSYS Customer Portal (<http://support.ansys.com>) and select the appropriate option.

**Telephone:** Please have your Customer or Contact ID ready.

UK: 0800 048 0462

Republic of Ireland: 1800 065 6642

Outside UK: +44 1235 420130

**Email:** support-uk@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

## JAPAN

### CFX and Mechanical Products

**Telephone:** +81-3-5324-7305

**Email:**

Mechanical: japan-ansys-support@ansys.com

Fluent: japan-fluent-support@ansys.com;

CFX: japan-cfx-support@ansys.com;

Polyflow: japan-polyflow-support@ansys.com;

### Icepak

**Telephone:** +81-3-5324-7444

**Email:** japan-icepak-support@ansys.com

### Licensing and Installation

**Email:** japan-license-support@ansys.com

## INDIA

### All ANSYS Products

**Web:** Go to the ANSYS Customer Portal (<http://support.ansys.com>) and select the appropriate option.

**Telephone:** +91 1 800 209 3475 (toll free) or +91 20 6654 3000 (toll) (Please have your Customer or Contact ID ready.)

## FRANCE

### All ANSYS Products

**Web:** Go to the ANSYS Customer Portal (<http://support.ansys.com>) and select the appropriate option.

**Toll-Free Telephone:** +33 (0) 800 919 225 **Toll Number:** +33 (0) 170 489 087 (Please have your Customer or Contact ID ready.)

**Email:** support-france@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

## BELGIUM

### All ANSYS Products

**Web:** Go to the ANSYS Customer Portal (<http://support.ansys.com>) and select the appropriate option.

**Toll-Free Telephone:** (0) 800 777 83 **Toll Number:** +32 2 620 0152 (Please have your Customer or Contact ID ready.)

**Email:** support-benelux@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

## SWEDEN

### All ANSYS Products

**Web:** Go to the ANSYS Customer Portal (<http://support.ansys.com>) and select the appropriate option.

**Telephone:** +46 (0) 10 516 49 00

**Email:** support-sweden@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

## SPAIN and PORTUGAL

### All ANSYS Products

**Web:** Go to the ANSYS Customer Portal (<http://support.ansys.com>) and select the appropriate option.

**Spain: Toll-Free Telephone:** 900 933 407 **Toll Number:** +34 9178 78350 (Please have your Customer or Contact ID ready.)

**Portugal: Toll-Free Telephone:** 800 880 513 (Portugal)

**Email:** support-spain@ansys.com, support-portugal@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

## ITALY

### All ANSYS Products

**Web:** Go to the ANSYS Customer Portal (<http://support.ansys.com>) and select the appropriate option.

**Toll-Free Telephone:** 800 789 531 **Toll Number:** +39 02 00621386 (Please have your Customer or Contact ID ready.)

**Email:** support-italy@ansys.com

Support for University customers is provided only through the ANSYS Customer Portal.

## TAIWAN, REPUBLIC OF CHINA

**Telephone:** 866 22725 5828

**KOREA**

**Telephone:** 82-2-3441-5000

**CHINA**

**Toll-Free Telephone:** 400 819 8999 **Toll Number:** +86 10 82861715



---

---

## **Part I: ANSYS Fluent Adjoint Solver**

Using This Manual (p. iii)

1. Introduction to the Adjoint Solver (p. 5)
2. Using the Adjoint Solver Module (p. 19)
3. Tutorial: 2D Laminar Flow Past a Cylinder; (p. 63)

---

---

---

# Using This Manual

---

## 1. The Contents of This Manual

The ANSYS Fluent Adjoint Solver Module Manual provides information about using the adjoint solver with ANSYS Fluent. An adjoint solver is a specialized tool that extends the scope of the analysis provided by a conventional flow solver by providing detailed sensitivity data for the performance of a fluid system. This part is divided into the following chapters:

- Introduction to the Adjoint Solver (p. 5)
- Using the Adjoint Solver Module (p. 19)
- Tutorial: Using the Adjoint Solver – 2D Laminar Flow Past a Cylinder (p. 63)



---

# Chapter 1: Introduction to the Adjoint Solver

---

This chapter provides background for the ANSYS Fluent adjoint solver.

- 1.1. Overview
- 1.2. Discrete Versus Continuous Adjoint Solver
- 1.3. Discrete Adjoint Solver Overview
- 1.4. Adjoint Solver Stabilization
- 1.5. Solution-Based Adaption
- 1.6. Using The Data To Improve A Design

In addition, see [Using the Adjoint Solver Module \(p. 19\)](#) and the Fluent Tutorial Guide for more information about using the adjoint solver. (To access tutorials and their input files on the ANSYS Customer Portal, go to <http://support.ansys.com/training>.)

## 1.1. Overview

An adjoint solver is a specialized tool that extends the scope of the analysis provided by a conventional flow solver by providing detailed sensitivity data for the performance of a fluid system.

In order to perform a simulation using the ANSYS Fluent standard flow solvers, a user supplies system geometry in the form of a computational mesh, specifies material properties and physics models, and configures boundary conditions of various types. The conventional flow solver, once converged, provides a detailed data set that describes the flow state governed by the flow physics that are being modeled. Various postprocessing steps can be taken to assess the performance of the system.

If a change is made to any of the data that defines the problem, then the results of the calculation can change. The degree to which the solution changes depends on how sensitive the flow is to the particular parameter that is being adjusted. Indeed, the derivative of the solution data with respect to that parameter quantifies this sensitivity to first order. Determining these derivatives is the domain of *sensitivity analysis*.

There is a large collection of derivative data that can be computed for a fluid system, given the extensive set of input data that is required, and the extensive flow data that is produced. The matrix of derivatives of output data with respect to input data can be vast. Depending upon the goal of the analysis only a portion of this derivative data may be needed for engineering analysis and decision-making.

The adjoint solver accomplishes the remarkable feat of calculating the derivative of a single engineering observation with respect to a very large number of input parameters simultaneously via a *single computation*. The engineering observation could be a measure of the system performance, such as the lift or drag on an airfoil, or the total pressure drop through a system. Most importantly, the derivatives with respect to the geometric shape of the system are found.

Understanding such sensitivities in a fluid system can provide extremely valuable engineering insight. A system that is highly sensitive may exhibit strong variability in performance due to small variations in manufacturing or variations in the environment in which it is operating. Alternatively, high-sensitivity may be leveraged for fluid control, with a small actuator being able to induce strong variations in behavior. Yet another perspective is that sensitivity of a performance measure implies that the device in

question is not fully optimized and there is still room for improvement—assuming that constraints do not preclude further gains.

The sensitivities of a fluid system provided by an adjoint solver satisfy a central need in gradient-based shape optimization. This makes an adjoint solver a unique and powerful engineering tool for design optimization.

Adjoint data can also play a role in improving solver numerics. Regions of high sensitivity are indicative of areas in the flow where discretization errors can potentially have a strong effect. This information can be used to guide how best to refine a mesh to improve flow solution accuracy.

The process of computing an adjoint solution resembles that for a standard flow calculation in many respects. The adjoint solver solution advancement method is specified, residual monitors configured, and the solver is initialized and run through a sequence of iterations to convergence. One notable difference is that a scalar-valued observation is selected as being of interest prior to starting the adjoint calculation.

Once the adjoint solution is converged the derivative of the observable with respect to the position of each and every point on the surface of the geometry is available, and the sensitivity of the observation to specific boundary condition settings can be found. This remarkable feature of adjoint solutions has been known for hundreds of years, but only in the last 25 years has the significance for computational physics analysis been recognized widely.

The power of this methodology is highlighted when an alternative for assembling the same information is considered. Imagine a sequence of flow calculations in which each point on an airfoil surface is moved in turn a set small distance in the surface-normal direction, and the flow and drag recomputed. If there are  $N$  points on the surface, then  $N$  flow calculations are required to build the data set. Considering that the same data is provided by a *single* adjoint computation, the adjoint approach has an enormous advantage. Remember that for even modest 3D flow computations there may be many thousands of coordinates or more on a surface.

Once the adjoint is computed it can be used to guide intelligent design modifications to a system. After all, the adjoint sensitivity data provides a map across the entire surface of the geometry of the effect of moving the surface. Design modifications can be most effective if made in regions of high sensitivity since small changes will have a large effect upon the engineering quantity of interest. This principle of making changes to a system in proportion to the local sensitivity is the foundation for the simple gradient algorithm for design optimization.

Once a candidate change in shape or other boundary condition has been selected, the effect of that change can be estimated using the computed derivative data. This amounts to a first order extrapolation using a Taylor series expansion around the baseline flow state. Clearly if a modification is chosen that is large enough that nonlinear effects become important then the accuracy of the predicted change cannot be guaranteed.

### 1.1.1. General Observables

Several observables are available and serve as the foundation for specifying the quantity that is of interest for the computation. Basic quantities such as forces and moments can be defined and each is given a name. The following types of observables are available:

- Force: the aerodynamic force in a specified direction on one or more walls.

- Moment of force: the aerodynamic moment about a specified moment center and moment axis. The integration is made over a specified collection of wall zones. Note that in 2D, the moment axis is normal to the plane of the flow.
- Swirl: the moment of the mass flow (with velocity  $\underline{u}$ ) relative to an axis defined by a point,  $\underline{r}_c$ , and a direction  $\underline{d}$ . The two options are:
  - Swirl integral

$$\int_V \rho(\underline{r} \times \underline{u}) \cdot \underline{d} dV \quad (1.1)$$

where  $V$  denotes the volume over which the integration is made, and  $\underline{r}$  denotes the relative position to the point  $\underline{r}_c$ .

- Normalized swirl integral

$$\frac{\int_V \rho(\underline{r} \times \underline{u}) \cdot \underline{d} dV}{\int_V \rho(\underline{r} \times (\underline{d} \times \underline{r})) \cdot \underline{d} dV} \quad (1.2)$$

### Note

These integrals can be used to construct quantities such as tumble ratio that are of significance for internal combustion engine analysis.

- pressure drop between an inlet (or group of inlets) and an outlet (or group of outlets)
- fixed value: a simple fixed value can be specified and used in the assembly of the observables. This is convenient when some scaling or normalization of the observable is desired. It must be noted that this value is treated strictly as a constant for the purposes of any derivative calculations.
- surface integral: a variety of surface integrals can be constructed for a specified field variable on a set of user-selected surfaces:

- Facet sum: a simple sum of the field value on each facet of the computational mesh

$$\sum_f \varphi_f \quad (1.3)$$

- Facet average: the facet sum is divided by the total number of facets,  $N_f$

$$\bar{\varphi} = \frac{\sum_f \varphi_f}{N_f} \quad (1.4)$$

- Facet variance: the sum of the squares of the deviations from the facet average

$$\frac{\sum_f (\varphi_f - \bar{\varphi})^2}{N_f} \quad (1.5)$$

- Integral: the sum of the field value on each face multiplied by the face area

$$\sum_f |A_f| \varphi_f \quad (1.6)$$

- Area-weighted average: the integral divided by the total face area

$$\frac{\sum_f |A_f| \varphi_f}{\sum_f |A_f|} \quad (1.7)$$

- Area-weighted variance: the sum of the squares of the deviations from the area-weighted average, divided by the total face area

$$\frac{\sum_f |A_f| (\varphi_f - \bar{\varphi})^2}{\sum_f |A_f|} \quad (1.8)$$

- Mass-weighted integral: the sum of the field value on each face, weighted by the magnitude of the local mass flow through the face.

$$\sum_f |\rho \underline{u}_f \cdot A_f| \varphi_f \quad (1.9)$$

- Mass-weighted average integral: the mass-weighted integral divided by the sum of the magnitudes of the local mass flow rates through the faces on which the integral is defined.

$$\frac{\sum_f |\rho \underline{u}_f \cdot A_f| \varphi_f}{\sum_f |\rho \underline{u}_f \cdot A_f|} \quad (1.10)$$

- Mass-weighted variance: the mass-weighted integral of the square of the deviation from the mass-weighted average, divided by the sum of the magnitudes of the local mass flow rates through the faces on which the integral is defined.

$$\frac{\sum_f |\rho \underline{u}_f \cdot A_f| (\varphi_f - \bar{\varphi})^2}{\sum_f |\rho \underline{u}_f \cdot A_f|} \quad (1.11)$$

- Flow-rate weighted: the rate at which the field is convected through the defined surfaces.

$$\sum_f (\rho \underline{u}_f \cdot A_f) \varphi_f \quad (1.12)$$

- The field variables that can be used are:

→ Pressure

→ Total pressure

→ Mass flow per unit area

→ Temperature (when adjoint energy is enabled)

- Heat flux (when adjoint energy is enabled)
- volume integral: you can compute a variety of volume integrals of a chosen field variable.
  - Volume: computes the total volume of the selected zones as a summation of the individual cell volumes.

$$\int dV = \sum_{i=1}^n |V_i| \quad (1.13)$$

- Sum: computes the sum of a chosen field variable over the selected zones.

$$\sum_{i=1}^n |\varphi_i| \quad (1.14)$$

- Volume Integral: is calculated by summing the product of cell volume and the selected field variable over the selected zones.

$$\int \varphi dV = \sum_{i=1}^n \varphi_i |V_i| \quad (1.15)$$

- Volume-Weighted Average: is computed by dividing the volume integral of a selected field variable ([Equation 1.15 \(p. 9\)](#)) by the total volume of the selected zones.

$$\int \varphi dV = \frac{1}{V} \sum_{i=1}^n \varphi_i |V_i| \quad (1.16)$$

- Volume Variance: is the volume-weighted integral of the square of the deviation from the volume-weighted average ([Equation 1.16 \(p. 9\)](#)), divided by the total volume of the selected zone.

$$\frac{\sum_{i=1}^n |V_i| (\varphi_i - \bar{\varphi})^2}{\sum_{i=1}^n |V_i|} \quad (1.17)$$

- Mass Integral: is computed by summing the product of density, cell volume, and the chosen field variable.

$$\int \rho \varphi dV = \sum_{i=1}^n \rho_i \varphi_i |V_i| \quad (1.18)$$

- Mass: is computed by summing the product of cell density and cell volume.

$$\int \rho dV = \sum_{i=1}^n \rho_i |V_i| \quad (1.19)$$

- Mass-Weighted Average: is computed by dividing the Mass Integral of the chosen field variable ([Equation 1.18 \(p. 9\)](#)) by the total Mass ([Equation 1.19 \(p. 9\)](#))

$$\frac{\int \rho \varphi dV = \sum_{i=1}^n \rho_i \varphi_i |V_i|}{\int \rho dV = \sum_{i=1}^n \rho_i |V_i|} \quad (1.20)$$

- Mass Variance: is computed as the mass-weighted summation of the square of the deviation of the chosen field variable from its mass-weighted average ([Equation 1.20 \(p. 9\)](#)) divided by the total mass ([Equation 1.19 \(p. 9\)](#)).

$$\frac{\sum_{i=1}^n \rho_i |V_i| (\varphi_i - \bar{\varphi})^2}{\sum_{i=1}^n \rho_i |V_i|} \quad (1.21)$$

The volume integrals listed above can be computed for the following field variables:

- Pressure
- Total Pressure
- Velocity
- Velocity Magnitude
- Vorticity
- Vorticity Magnitude
- Turbulence Production:  $\frac{\partial u_i}{\partial x_j} \left( \frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$

### 1.1.2. General Operations

Several operations are available to combine observables in various ways, if needed, to make a wide variety of compound observables. The following operations are available:

- ratio of two quantities
- product of two quantities
- linear combination of  $N$  quantities  $q_0 \dots q_{N-1}$ , with constant coefficients  $a_0 \dots a_{N-1}$  raised to various powers  $v_0 \dots v_{N-1}$ , plus a constant offset  $c$ .

$$c + \sum_{i=0}^{N-1} a_i q_i^{v_i} \quad (1.22)$$

- arithmetic average of  $N$  quantities  $q_0 \dots q_{N-1}$

$$\bar{q} = \frac{\sum_{i=0}^{N-1} q_i}{N} \quad (1.23)$$

- mean variance of  $N$  quantities

$$\frac{\sum_{i=0}^{N-1} (q_i - \bar{q})^2}{N} \quad (1.24)$$

- unary operation (sin, cosine, and so on) on observable value  $q$ :

- $1/q$
- $|q|$

- $\sqrt{q}$
- $\sin q$
- $\cos q$
- $\tan q$
- $\sin^{-1} q$
- $\cos^{-1} q$
- $\tan^{-1} q$
- $\ln q$
- $\log_{10} q$

## 1.2. Discrete Versus Continuous Adjoint Solver

When an adjoint solver is developed there is a key design decision that is made regarding the implementation, namely whether to use a *continuous* or a *discrete* approach. Both approaches are intended to compute the same sensitivity data—however, the approaches are remarkably different.

While a user of the adjoint solver may experience no particular difference in workflow for the two separate solver types, it is important to be aware that there are two primary classes of adjoint solver and the approach chosen can have implications for the accuracy of the results.

A *continuous adjoint solver* relies heavily on mathematical properties of the partial-differential equations that define the physics of the problem. In this case those equations are the Navier-Stokes equations. With this approach an adjoint partial differential equation set is formulated explicitly and is accompanied by adjoint boundary conditions that are also derived mathematically. Only after this derivation is complete can the adjoint partial differential equations be discretized and solved, often with extensive re-use of existing solver machinery. This class of solver was implemented by ANSYS Fluent as a research effort.

Such a solver has the benefit that it is decoupled largely from the original flow solver. They share only the fact that they are based on the Navier-Stokes equations. The process of discretizing and solving the partial differential equations in each case could in principle be very different indeed. While this flexibility may be appealing it can also be the downfall of the approach. Inconsistencies in modeling, discretization and solution approaches can pollute the sensitivity information significantly, especially for problems with wall functions and complex engineering configurations such as those of interest to ANSYS Fluent users.

The continuous adjoint approach can be effective for some classes of problems. However, until there is a significant advance in handling some key challenges, ANSYS Inc. has concluded that it is an unsuitable approach for meeting the needs of our broad client base for the classes of problems of interest to them.

A *discrete adjoint solver* is based not on the form of the partial differential equations governing the flow, but the particular discretized form of the equations used in the flow solver itself. The sensitivity of the discretized equations forms the basis for the sensitivity calculation. In this approach the adjoint solver is much more tightly tied to the specific implementation of the original flow solver. This has been observed to yield sensitivity data that provides valuable engineering guidance for the classes of problem of interest to ANSYS Fluent users, including problems with wall functions.

For the above reasons, the discrete adjoint approach has been adopted for the ANSYS Fluent adjoint solver.

## 1.3. Discrete Adjoint Solver Overview

As discussed in the previous section, ANSYS, Inc. has chosen to adopt a discrete adjoint approach to solving the adjoint problem since it is believed to provide the most useful engineering sensitivity data for the classes of problems of interest to ANSYS clients.

An adjoint method can be used to compute the derivative of an observation of interest for the fluid system with respect to all the user-specified parameters, *with any changes that arise in the flow variables themselves eliminated*. There are three key ingredients to consider when developing the method:

### 1. All of the user-specified inputs:

- All values set by a user in the boundary condition panels for each boundary in the problem.
- The computational mesh. More specifically the locations of the nodes of the mesh and how they define the edges, faces, and ultimately the cells used in the finite-volume computation. This includes both interior as well as boundary nodes.
- Material properties.
- Model parameters such as model coefficients for turbulence models.

Note that the settings that define the problem are being distinguished from settings that define how the solution advancement is to be performed to converge the problem. Only the former are of interest here. For the sake of clarity, let us denote the vector of all of the values in the list by  $\underline{c}$ . These are considered to be the control variables for the problem, that is, the variables that a user can set explicitly that affect the solution.

It is worth noting that the topological definition of the mesh is fixed—it is not considered to be a control variable here. The effect of collapsing cells or remeshing on the flow solution is not addressable without further development in view of the discrete changes that are implied. This topic is beyond the scope of the current document.

### 2. The governing equations for the fluid system:

The main effort in a flow computation is in the determination of the flow state, namely the velocity, pressure, density and possibly other fluid-related variables. For a cell-centered finite-volume scheme, the flow state is defined at the cell centroids by a vector of real values. In the simplest case these values are the pressure and flow velocity components. Let the vector of the variables in the  $v^{\text{th}}$  cell be denoted here by  $\underline{q}^v$ .

At convergence the flow variables satisfy

$$\mathcal{R}_i^{\mu}(\underline{q}^0, \underline{q}^1, \dots, \underline{q}^{M-1}; \underline{c}) = 0, \quad \mu = 0, \dots, M-1, \quad i = 0, \dots, L-1 \quad (1.25)$$

where  $M$  is the number of cells in the problem, and there are  $L$  conditions on each cell. This expression is a compact way of denoting conservation of mass and momentum and other constraints.

### 3. The engineering observation of interest:

Let

$$\mathcal{J}(q^0, q^1, \dots, q^{M-1}; \underline{c}) \quad (1.26)$$

denote a scalar of interest that depends both on the flow state and perhaps directly on the control variables. It is assumed that the observable is differentiable with respect to both the flow and the controls. The inclusion of the control variables here is essential since in many cases the mesh geometry is included directly in the evaluation of the observable. For example, the evaluation of the force on a boundary involves the wall normal and face areas, which change when mesh nodes are moved.

The goal is to determine the sensitivity of the observation with respect to the user-specified control variables. What makes defining this relationship more challenging is the fact that changing the user inputs changes the flow, which indirectly changes the engineering observation. The adjoint method has a specific role in managing this chain of influences by providing a mechanism for eliminating the specific changes that happen in the flow whenever the inputs change.

If a variation  $\delta c_j$  is introduced into the control variables then a linearization of the governing equations ([Equation 1.25 \(p. 12\)](#)) shows that the variations in the flow state  $\delta q_j^\nu$  must satisfy

$$\frac{\partial R_i^\mu}{\partial q_j^\nu} \delta q_j^\nu = -\frac{\partial R_i^\mu}{\partial c_j} \Big|_q \delta c_j, \quad \mu=0,\dots,M-1, \quad i=0,\dots,L-1. \quad (1.27)$$

where there is an implied summation over  $j$  and  $\nu$ , and  $\Big|_q$  denotes that the flow solution is held constant while the derivative is taken.

Meanwhile, if both the control variables and the flow state change, then the observation will change:

$$\delta \mathcal{J} = \frac{\partial \mathcal{J}}{\partial q_j^\nu} \delta q_j^\nu + \frac{\partial \mathcal{J}}{\partial c_j} \Big|_q \delta c_j. \quad (1.28)$$

The particular way in which the flow responds to the changes in the control variables can be computed using ([Equation 1.27 \(p. 13\)](#)) only after specific changes,  $\delta c_j$ , have been chosen. It is prohibitive to consider solving ([Equation 1.27 \(p. 13\)](#)) for more than a handful of prescribed changes  $\delta c_j$  because of the excessive computing time that would be needed. However, when redesigning the shape of parts of a system there may be pressure to explore a large number of candidate modifications. This conflict is reconciled by eliminating the variations of the flow solution from the expression ([Equation 1.28 \(p. 13\)](#)) and producing an explicit relationship between changes in the control variables and the observation of interest.

This is accomplished by taking a weighted linear combination of the linearized governing equations ([Equation 1.27 \(p. 13\)](#)) in a very particular way. A set of adjoint variables  $\tilde{q}_i^\mu$  is introduced with a one-to-one correspondence with the governing equations ([Equation 1.25 \(p. 12\)](#)). This results in a relationship

$$\left[ \tilde{q}_i^\mu \frac{\partial R_i^\mu}{\partial q_j^\nu} \right] \delta q_j^\nu = -\tilde{q}_i^\mu \frac{\partial R_i^\mu}{\partial c_j} \Big|_q \delta c_j. \quad (1.29)$$

The term in square brackets on the left is now matched to the coefficient for the variation in the flow in ([Equation 1.28 \(p. 13\)](#)) in order to define values for the adjoint variables:

$$\frac{\partial R_i^\mu}{\partial q_j^\nu} \tilde{q}_i^\mu = \frac{\partial \mathcal{J}}{\partial q_j^\nu}. \quad (1.30)$$

These are the discrete adjoint equations, and the solution of this system that is the primary goal for the adjoint solver. It is important to recognize that these equations have not been derived. They are

defined in this way with a specific goal in mind—the elimination from (Equation 1.28 (p. 13)) of the perturbations to the flow field as shown below:

$$\begin{aligned}
 \delta\mathcal{J} &= \frac{\partial\mathcal{J}}{\partial q_j^\nu} \delta q_j^\nu + \left. \frac{\partial\mathcal{J}}{\partial c_j} \right|_q \delta c_j \\
 &= \tilde{q}_i^\mu \frac{\partial R_i^\mu}{\partial q_j^\nu} \delta q_j^\nu + \left. \frac{\partial\mathcal{J}}{\partial c_j} \right|_q \delta c_j \\
 &= \left\{ \left. \frac{\partial\mathcal{J}}{\partial c_j} \right|_q - \tilde{q}_i^\mu \frac{\partial R_i^\mu}{\partial c_j} \right\} \delta c_j
 \end{aligned} \tag{1.31}$$

Note the use of Equation 1.29 (p. 13) and Equation 1.30 (p. 13) in the derivation of Equation 1.31 (p. 14). The flow perturbation has now been eliminated from the expression, yielding a direct relation between the control variables and the observable of interest.

There are several important observations to be made about (Equation 1.30 (p. 13)):

- The dimension of the problem to be solved is the same as the original flow problem, although the adjoint problem is linear.
- While the adjoint solution may be considered strictly as a vector of numeric values, experience with the continuous adjoint provides guidance on how the adjoint solution can be interpreted. The vector of weights associated with the components of the residual of the momentum equation in each cell is termed the *adjoint velocity*. The adjoint value associated with the residual of the continuity equation is termed the *adjoint pressure*.
- The right hand side is defined purely on the basis of the observable that is of interest.
- The matrix on the left hand side is the *transpose* of the Jacobian of the governing system of equations (Equation 1.25 (p. 12)). This seemingly innocent transposition has a very dramatic impact on how the adjoint system is solved. This will be discussed below.
- The adjoint equations are defined by the current state of the flow, and the specific physics that is employed in the modeling. Each adjoint solution is specific to the flow state.

At first glance it appears that solving the adjoint problem may be straightforward. After all it simply involves setting-up and solving a linear (albeit large) system of equations. In practice both steps can represent a significant challenge, especially when the problem is large.

The evaluation of the residuals of the flow equations is an integral part of the pre-existing ANSYS Fluent flow solvers. However, it is necessary to compute the Jacobian of the system  $\partial R_i^\mu / \partial q_j^\nu$  and then transpose it, or at the very least to be able to make a matrix-free transpose matrix-vector product with the adjoint solution. There are several technical approaches to accomplishing this task whose description goes beyond the scope of this document. Suffice to say that it is not trivial to encode this functionality, but that it has been done successfully here.

For the present implementation, a pre-conditioned iterative scheme, based on pseudo-time marching, is adopted to solve the adjoint system. The equations that define the advancement process can be written as

$$\left\{ \frac{P_{ij}^{\mu,\nu}}{CFL\Delta t} + L_{ij}^{\mu,\nu} \right\} \Delta \tilde{q}_i^\nu = \frac{\partial\mathcal{J}}{\partial q_j^\mu} - \frac{\partial R_i^\mu}{\partial q_j^\nu} \tilde{q}_i^\nu \tag{1.32}$$

where  $\Delta t$  is a local time step size based on the local flow conditions,  $CFL$  is a user-specified  $CFL$  number. The matrix  $L_{ij}^{\mu,\nu}$  is a simplified form of the system Jacobian that is amenable to solution using the AMG linear solver that is the workhorse of the conventional flow solver. The preconditioning matrix  $P_{ij}^{\mu,\nu}$  is a diagonal matrix. Artificial compressibility is introduced on the adjoint continuity equation to aid in the relaxation of the adjoint pressure field.

The correction  $\Delta \tilde{q}_i^\mu$  is under-relaxed and added to the adjoint solution  $\tilde{q}_i^\mu$  and the process repeated until the right hand side is adequately small.

## 1.4. Adjoint Solver Stabilization

When applied to problems with large cell counts and complex geometry, adjoint solvers sometimes experience stability issues. These instabilities can be associated with small scale unsteadiness in the flow field and/or strong shear, and tend to be restricted to small and isolated regions of the flow domain. Despite the spatial localization of these instabilities, the linearity of the adjoint problem provides no intrinsic limit on their growth during solution advancement. Their presence, if not handled, can disrupt the entire adjoint calculation despite the problem occurring in sometimes just a few cells. A stabilized solution scheme will be required to obtain adjoint solutions for problems at high Reynolds number in which there is strong shear and/or complex geometry.

Three stabilization schemes are available in Fluent in order to overcome these stability difficulties when larger cases are being solved. The stabilization schemes are designed to intervene only when the standard advancement scheme is experiencing instability.

### Spatial scheme

The spatial scheme operates by identifying parts of the domain where unstable growth is occurring and applying a more direct and stable solution procedure in those regions.

### Modal scheme

The modal scheme involves a process of identifying the particular details of the unstable growth patterns or modes. These patterns are localized in space and they are used to split the solution into parts that have stable and unstable characteristics when advanced. The stable part is advanced as usual, while the algorithm is designed now to compensate for the unstable part so that the overall calculation is stabilized.

The emergence of the unstable patterns can happen at any time during the adjoint calculation and the total number of unstable patterns is case-dependent. Having 10 to 20 unstable modes present would not be considered unusual. Large cases may have many more. As a general trend, the most rapidly-growing unstable patterns appear often within a few iterations with more slowly-growing modes appearing later in the calculation. The unstable patterns are handled as they appear.

### Dissipation scheme

The dissipation scheme provides stabilization for the solution advancement of the adjoint solution by introducing nonlinear damping strategically into the calculation domain. The strategy is intended to provide minimal intervention in order to damp the growth of instabilities that lead to adjoint solution divergence. A marker is tracked, based on the state of the adjoint solution, and damping is applied directly to the adjoint solution in regions where the marker becomes large.

Unlike the modal and spatial schemes, the dissipation scheme can affect the adjoint solution slightly. In general, the spatial order of the damping is chosen to be one order larger than the adjoint calculation order. This means that the formal order of accuracy of the adjoint solver is unaffected by the addition of the dissipation scheme. In practice, regions of intense dissipation can appear for some cases which may result in isolated spots and/or streaks in the solution on surfaces. However, the

scale of these solution features is such that they are often easily smoothed during postprocessing to generate design changes.

Some additional computational overhead is associated with the stabilization schemes. In general, the dissipation scheme is the least resource-intensive of the three schemes.

For information about using the stabilization schemes, refer to [Stabilized Scheme Settings \(p. 32\)](#).

## 1.5. Solution-Based Adaption

An adjoint solution provides guidance on where best to adapt a computational mesh in order to resolve quantities of engineering interest.

Once the governing equations for the system ([Equation 1.25 \(p. 12\)](#)) have been converged there remains a discretization error,  $\varepsilon_i^\mu$ , such that

$$\mathcal{R}_i^\mu(q^0, q^1, \dots, q^{M-1}; \underline{c}) = \varepsilon_i^\mu, \quad \mu=0, \dots, M-1, \quad i=0, \dots, L-1. \quad (1.33)$$

While specific estimates for this discretization error may be tricky to define, it is often estimated to be  $O(h^p)$ , where  $h$  is the local grid size, and  $p$  is the order of the discretization scheme. That is,  $p=1$  for a first-order scheme and  $p=2$  for a second order scheme. Alternatively,  $\varepsilon_i^\mu$  can be considered to be the residual associated with a solution that is not converged fully.

The correction to the flow field,  $\delta q_j^\nu$ , that compensates for this inhomogeneity is given by

$$\frac{\partial R_i^\mu}{\partial q_j^\nu} \delta q_j^\nu = -\varepsilon_i^\mu \quad (1.34)$$

from which it follows quickly that

$$\delta \mathcal{J} = \frac{\partial \mathcal{J}}{\partial q_j^\nu} \delta q_j^\nu = -\varepsilon_i^\mu \hat{q}_i^\mu. \quad (1.35)$$

This simple expression provides an estimate of the effect of the presence of  $\varepsilon_i^\mu$  on the observation,  $\mathcal{J}$ .

The presence of discretization errors, or lack of convergence, on the engineering quantity of interest is assessed by weighting the inhomogeneous term by the local adjoint solution. It is clear that even in regions of the domain where the residuals or discretization errors are small, an accompanying adjoint velocity or pressure that is large in magnitude implies that there may be a significant source of error in the observable. A finer mesh in regions where the adjoint is large will reduce the influence of discretization errors that may adversely affect the engineering result of interest. In practice, adapting cells which have large magnitude adjoint velocity and/or adjoint pressure will achieve this goal.

## 1.6. Using The Data To Improve A Design

Adjoint sensitivity data can be used to guide how to modify a system in order to improve the performance. The observable of interest can be made larger or smaller, depending upon the engineering goal.

A common strategy for deciding how to modify the system is based on the gradient algorithm. The underlying principle is quite simply that modifying a system in a manner to which it is most sensitive maximizes the effect of the change. The change to a control variable is made in proportion to the sensitivity of the value of interest with respect to that control variable.

Denote the sensitivity of the cost with respect to shape by

$$\delta J = \frac{\partial J}{\partial x_j^n} \delta x_j^n \quad (1.36)$$

where  $x_j^n$  is the  $j^{\text{th}}$  coordinate of the  $n^{\text{th}}$  node in the mesh. Here  $x_j^n$  is a notation for the subset of the control variables  $c_j$  for the system that correspond to mesh node positions. Then an adjustment

$$\delta x_j^n = \lambda \frac{\partial J}{\partial x_j^n} \quad (1.37)$$

will provide the maximum adjustment to  $J$  for given  $L^2$  norm of  $\delta x_j^n$ , where  $\lambda$  is an arbitrary scaling factor. Note that  $\lambda$  can be picked to be positive or negative depending upon whether  $J$  is to be increased or decreased respectively. This is essentially a statement of the method of steepest descent.

Furthermore, the change is estimated to first order to be

$$\delta J = \lambda \frac{\partial J}{\partial x_j^n} \frac{\partial J}{\partial x_j^n}. \quad (1.38)$$

For a sufficiently small adjustment, the change to the observation will strictly have the same sign as the scaling factor  $\lambda$ , provided the gradient is not identically zero.

In regions where the sensitivity is high, small adjustments to the shape will have a large effect on the observable. This corresponds directly with the idea of engineering robustness. If a configuration shows high sensitivities, then the performance will likely be subject to large performance variations if there are manufacturing inconsistencies. For a robust design, the goal is to have the sensitivity be tolerably small.

In practical cases the field  $\partial J / \partial x_j^n$  can be noisy, especially for large, turbulent flow problems. If the noisy field is used directly to modify a boundary shape using (Equation 1.37 (p. 17)), then the modified surface can have many inflections. This is not helpful for engineering design work. In the next section, the use of mesh morphing technology not only to smooth the sensitivity field, but also to provide smooth boundary and interior mesh deformation, is described.

### 1.6.1. Smoothing and Mesh Morphing

As has been noted, for typical engineering problems, the shape sensitivity field can have smoothness properties that are not adequate to define a shape modification. Mesh morphing technology is used here for both two- and three-dimensional systems in a two-fold role. The first role is as a smoother for the surface sensitivity field. The second role is to provide smooth distortions not only of the boundary mesh, but also the interior mesh. This approach is very appealing since it functions for arbitrary mesh cell types.

#### Note

For simplicity, the smoothing and morphing equations are presented here for the case of a 2D domain. The same approach is extended and used for 3D problems.

A Cartesian region is selected that encompasses all or a part of the problem domain. Only the mesh nodes that fall within this region may move as a result of the morphing operation. A local  $(u, v)$  coordinate system is defined, where  $u \in [0, 1]$  and  $v \in [0, 1]$ . A regular array of  $N_u \times N_v$  control points is then distrib-

uted in the control volume. The motion of the mesh at any point in the domain is determined by the movement of the control points.

In general, an optimization problem may include both free-form minimization or maximization of the observable(s) and constraints that are localized in space. This requires an approach that can provide a deformation field that is well-behaved and consistent with manufacturability requirements, while still allowing locally sharper deformations where required to satisfy the imposed constraints.

To achieve this, two coincident sets of control points are created. Bernstein polynomials and B-splines are used, respectively, to map the control point motions to the computational mesh nodes.

The  $i^{\text{th}}$  Bernstein polynomial of degree  $l$  is:

$$B_{i,l}(u) = \binom{l}{i} u^i (1-u)^{l-i} \quad (1.39)$$

The  $i^{\text{th}}$  B-spline of degree  $p$  is the non-periodic B-spline that uses a knot-vector,  $\{t_i\}$ , where:

$$t_i = \begin{cases} 0 & i \leq p \\ \frac{(i-p)}{(l-2p)} & p < i < l-p \\ 1 & i \geq l-p \end{cases} \quad (1.40)$$

The value of the B-spline is defined recursively as:

$$\begin{aligned} S_{i,0}(t) &= \begin{cases} 1 & t_i < t < t_{i+1}, t_i < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \\ S_{i,j}(t) &= \frac{t-t_i}{t_{i+j}-t_i} S_{i,j-1}(t) + \frac{t_{i+j+1}-t}{t_{i+j+1}-t_{i+1}} S_{i+1,j-1}(t) \end{aligned} \quad (1.41)$$

Depending on the degree of the B-spline, the function is zero on some portion of the unit interval. This is in contrast to the Bernstein polynomials that are strictly non-zero everywhere on the unit interval.

Let  $x_i^v$  denote the  $i^{\text{th}}$  coordinate of the  $v^{\text{th}}$  mesh node, and let  $\Delta x_i^v$  denote the change in the position of the node due to the morphing process. Let  $\Delta \eta_{jk}^i$  denote the displacement of the  $(j,k)^{\text{th}}$  control point associated with the Bernstein polynomials in the  $i^{\text{th}}$  coordinate direction, and let  $\Delta \zeta_{jk}^i$  denote the displacement of the  $(j,k)^{\text{th}}$  control point associated with the B-splines. If  $(u^v, v^v)$  denotes the position of the  $v^{\text{th}}$  mesh node, then the displacement of the node is defined by the superposition:

$$\Delta x_i^v = \sum_{j=0}^{N_u} \sum_{k=0}^{N_v} \left( \Delta \eta_{jk}^i B_{j,l}(u^v) B_{k,m}(v^v) + \Delta \zeta_{jk}^i S_{j,l}(u^v) S_{k,m}(v^v) \right) \quad (1.42)$$

The control-point movement for the Bernstein polynomials controls the large-scale smooth deformation, while the control-point movement for the B-splines controls fine-scale motions.

---

---

## Chapter 2: Using the Adjoint Solver Module

---

This chapter describes the process for loading the adjoint solver module, as well as setting up, running, and postprocessing the adjoint solutions. Also, this chapter demonstrates the shape modification process that is guided by the adjoint solution.

The typical use of the adjoint solver involves the following steps:

1. Load or compute a conventional flow solution.
2. Load the adjoint solver module.
3. Specify the observable of interest.
4. Set the adjoint solver controls.
5. Set the adjoint solver monitors and convergence criteria.
6. Initialize the adjoint solution and iterate to convergence.
7. Post-process the adjoint solution to extract the sensitivity of the observable with respect to boundary condition settings.
8. Post-process the adjoint solution to extract the sensitivity of the observable with respect to shape of the geometry.
9. Modify boundary shapes based on shape-sensitivity data and recompute the flow solution.

This chapter provides information about using the ANSYS Fluent adjoint solver in the following sections:

- 2.1. [Installing the Adjoint Solver Module](#)
- 2.2. [Loading the Adjoint Solver Module](#)
- 2.3. [Model Considerations for Using Adjoint Solver](#)
- 2.4. [Defining Observables](#)
- 2.5. [Solving the Adjoint](#)
- 2.6. [Postprocessing of Adjoint Solutions](#)
- 2.7. [Modifying the Geometry Using the Design Tool](#)
- 2.8. [Using the Adjoint Solver Module's Text User Interface](#)

In addition, see the Fluent Tutorial Guide for more information about using the adjoint solver. (To access tutorials and their input files on the ANSYS Customer Portal, go to <http://support.ansys.com/training>.)

### 2.1. Installing the Adjoint Solver Module

The adjoint solver model is provided as an add-on module with the standard ANSYS Fluent licensed software. The module is installed with the standard installation of ANSYS Fluent in a directory called `addons/adjoint` in your installation area. The adjoint solver module consists of a UDF library and a pre-compiled scheme library, which need to be loaded and activated before calculations can be performed.

## 2.2. Loading the Adjoint Solver Module

The adjoint solver module is loaded into ANSYS Fluent through the text user interface (TUI). The module can be loaded only when a valid ANSYS Fluent case file has been set or read. The text command to load the module is

```
define → models → addon-module
```

A list of ANSYS Fluent add-on modules is displayed:

The adjoint solver module is loaded as follows:

```
> /define/models/addon-module
Fluent  Addon Modules:
 0. None
 1. MHD Model
 2. Fiber Model
 3. Fuel Cell and Electrolysis Model
 4. SOFC Model with Unresolved Electrolyte
 5. Population Balance Model
 6. Adjoint Solver
 7. Single-Potential Battery Model
 8. Dual-Potential MSMD Battery Model
 9. PEM Fuel Cell Model
Enter Module Number: [0] 6
```

Select the adjoint solver model by entering the module number 6. During the loading process a scheme library containing the graphical and text user interface, and a UDF library containing a set of user-defined functions (UDFs) are loaded into ANSYS Fluent. The **Adjoint-Based** group box in the **Design** ribbon tab becomes available once the adjoint module is loaded.

## 2.3. Model Considerations for Using Adjoint Solver

The current adjoint solver implementation provides basic adjoint solutions that accompany a conventionally-computed flow solution provided certain criteria are met.

### 2.3.1. Basic Assumptions and Consistency Checks

#### 2.3.2. User-Defined Sources

### 2.3.1. Basic Assumptions and Consistency Checks

The adjoint solver is implemented on the following basis:

- The flow state is for a steady single-phase flow, or ideal gas, in an inertial reference frame that is either laminar or turbulent.
- For turbulent flows a *frozen turbulence* assumption is made, in which the effect of changes to the state of the turbulence is not taken into account when computing sensitivities.
- For turbulent flows standard wall functions are employed on all walls.
- The adjoint solver uses methods that are first order accurate in space by default. If desired, you can select second order accurate methods (see [Using the Adjoint Solution Methods Dialog Box \(p. 29\)](#)).
- The boundary conditions are only of the following types:
  - Wall
  - Velocity inlet

- Pressure outlet
- Symmetry
- Rotational and translation periodic

It is important to note that these requirements are not strict conditions for the conventional flow solver, but rather modeling limitations for the adjoint solver.

When the adjoint solver is initialized or an observable is evaluated, and before iterations are performed, a series of checks is performed to determine the suitability of the existing flow solution for analysis with the adjoint solver. Two types of message may appear:

- *Warning message:* There are certain physics models and boundary condition types that are not explicitly modeled in the adjoint solver, but their absence does not disallow you from proceeding with the calculation. In this case, a warning message will be printed that explains the nature of the inconsistency.

```
Checking adjoint setup...
-- Warning: Model is active but not included in adjoint calculation: P1 radiation model Done
```

The adjoint solver will still run in this case but the quality of the adjoint solution data can be expected to be poorer as a result of the inconsistency. This is because in such cases the unsupported settings will revert to corresponding supported settings for the adjoint solver. Though the calculation will proceed, it is important to note that the results produced should be considered on this basis. When reverting back to the fluid calculation, the original settings will be preserved and the modified settings are not migrated onto the original case. The warning indicates that the setting change will be made automatically and specifically for the adjoint solution.

- *Error message:* There are some model and boundary conditions that are incompatible with the computation of an adjoint solution with the current adjoint solver implementation. In this case, an explanatory error message will be printed in the console window.

```
Checking adjoint setup...
** Unable to proceed: Model is active but not compatible with adjoint calculation:
Transition SST turbulence model
Done
```

The adjoint solver will not run in this case and changes must be made manually to the settings identified in the console before the solver can be run.

### 2.3.2. User-Defined Sources

User-defined sources for the momentum equations that are defined in compiled UDFs can be accounted for in the application of the adjoint solver, provided that the source derivatives with respect to flow variables and cell centroid coordinates are provided. This is achieved by creating the source term UDF definition in the usual fashion, with the exception that a special UDF macro is used, `DEFINE_SOURCE_AE` (The suffix `_AE` denotes Adjoint Enabled).

`DEFINE_SOURCE_AE (name, c, t, dS, eqn)`

Argument Type	Description
<code>symbol name</code>	UDF name
<code>cell_t c</code>	Index that identifies cell on which the source term is to be applied
<code>Thread *t</code>	Pointer to cell thread

**Argument Type**

real ds[ ]

int eqn

**Description**

Array that contains the derivative of the source term with respect to the dependent variable of the transport equation, as well as the flow variables and cell centroid position

Equation number

**Function returns**

real

Consider a source that may depend on velocity, pressure, and cell-centroid coordinates:

$$S(u, v, w, p, x_c, y_c, z_c)$$

where  $u$ ,  $v$ , and  $w$  are the Cartesian velocity components in the cell,  $p$  is the pressure in the cell, and  $(x_c, y_c, z_c)$  are the cell centroid coordinates as defined by the C\_CENTROID macro. If the source strength depends on one or more of these quantities, then the derivatives of the source strength with respect to those variables must be computed and filled in as entries to the `ds[ ]` array. In particular,

$$dS[D_U] = \frac{\partial S}{\partial u}$$

$$dS[D_V] = \frac{\partial S}{\partial v}$$

$$dS[D_W] = \frac{\partial S}{\partial w}$$

$$dS[D_P] = \frac{\partial S}{\partial p}$$

$$dS[D_X_CENTROID] = \frac{\partial S}{\partial x_c}$$

$$dS[D_Y_CENTROID] = \frac{\partial S}{\partial y_c}$$

$$dS[D_Z_CENTROID] = \frac{\partial S}{\partial z_c}$$

must be defined in the UDF if there is a dependency of the source upon them. The specification of this derivative information is essential for sensitivities, including shape sensitivity, to be computed correctly by the adjoint solver.

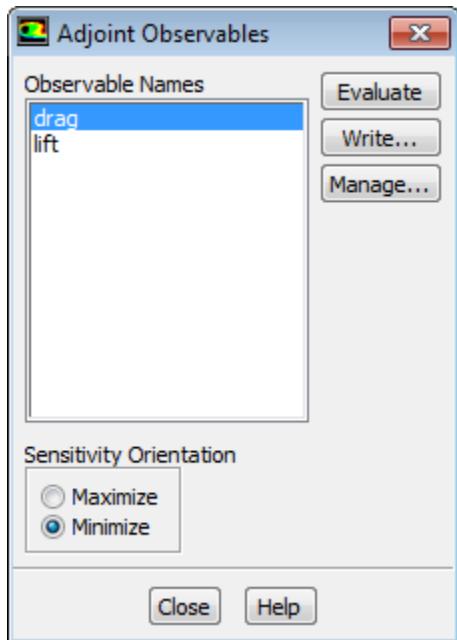
## 2.4. Defining Observables

Defining an observable is a key initial step that must be performed for any adjoint calculation. The observable is the quantity for which sensitivities are sought. Observables can be defined based on flow variables, or as operations on other observables. For a list of the types of observables you can define refer to [General Observables \(p. 6\)](#) and [General Operations \(p. 10\)](#). You can define multiple observables, but only one observable at a time can be selected for the adjoint sensitivity calculation.

Observables are created and selected in the **Adjoint Observables** dialog box, which is accessed by clicking **Observable...** in the **Design** ribbon tab (**Adjoint-Based** group box). (Figure 2.1: Adjoint Observables Dialog Box (p. 23)).



**Design** → **Adjoint-Based** → **Observable...**

**Figure 2.1: Adjoint Observables Dialog Box**

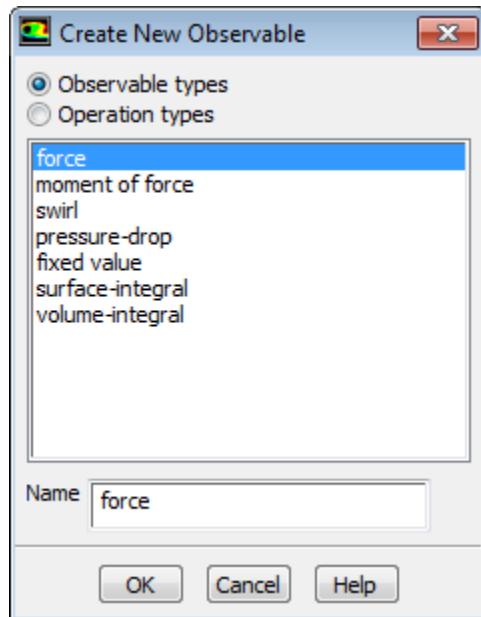
For more information, see the following sections:

- [2.4.1. Creating New Observables](#)
- [2.4.2. Editing Observable Definitions](#)
- [2.4.3. Selecting an Observable for Sensitivity Calculation](#)

## 2.4.1. Creating New Observables

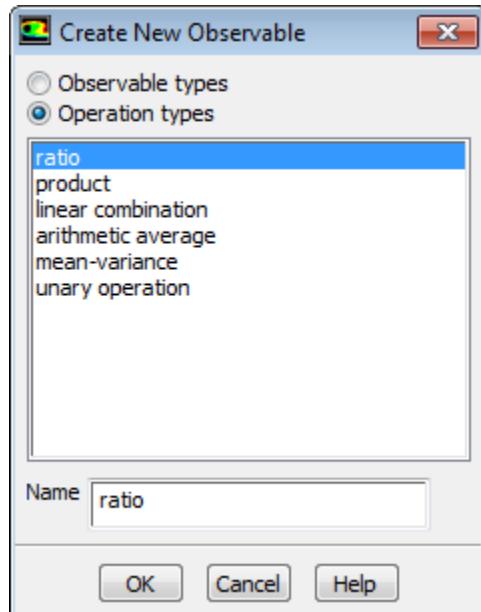
You can create a new observable by clicking **Manage...** in the **Adjoint Observables** dialog box and then **Create...** in the **Manage Adjoint Observables** dialog box.

In the **Create New Observable** dialog box (Figure 2.2: Create New Observable Dialog Box (Observable Types) (p. 24)) you can choose from several observable types and operation types. Various types of observable quantities can be defined and each can be given a name. The available types of observables are described in [General Observables](#) (p. 6).

**Figure 2.2: Create New Observable Dialog Box (Observable Types)**

Select **Observable types** and pick an observable from the list. Keep the default name, or use the **Name** field to designate a different name for the observable of interest. Click **OK** to create the new observable, or click **Cancel** to dismiss the dialog box without creating an observable.

You can also combine existing observables in various ways or apply unary operations to create a wide variety of compound observables. The operations that you can apply to existing observables are described in [General Operations \(p. 10\)](#).

**Figure 2.3: Create New Observable Dialog Box (Operation Types)**

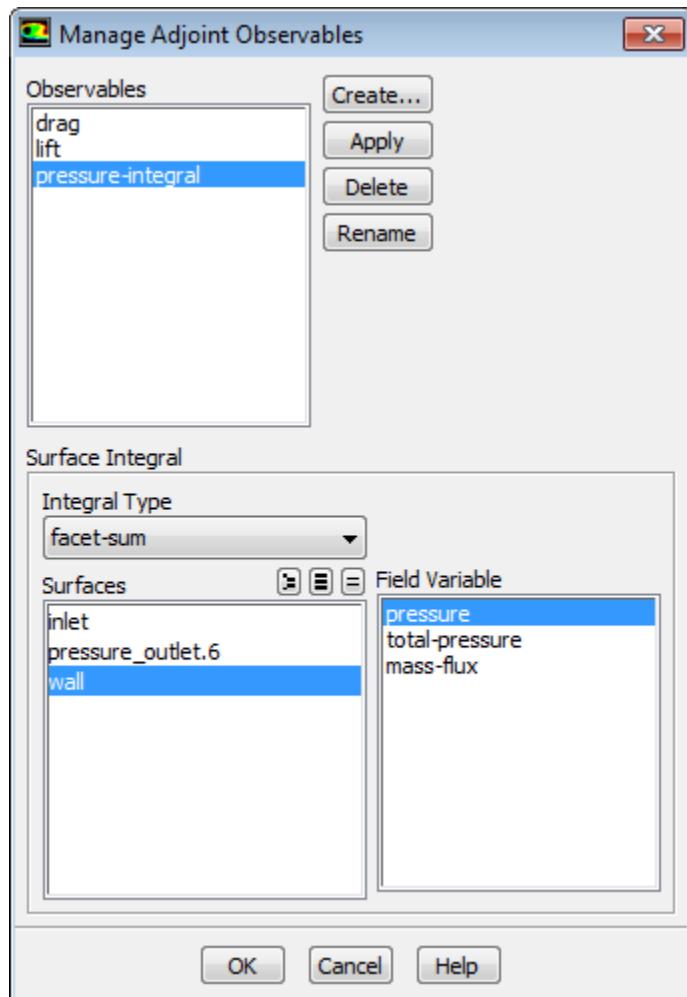
Select **Operation types** and choose an operation from the list. Keep the default name, or use the **Name** field to designate a different name for the observable of interest. Click **OK** to create the new observable, or click **Cancel** to dismiss the dialog box without creating an observable.

Once created, you must edit the definition of the new observable ([Editing Observable Definitions \(p. 25\)](#)).

## 2.4.2. Editing Observable Definitions

Once an observable is created, it appears in the **Observables** list. When you select an observable in the **Observables** list, the **Manage Adjoint Observables** dialog box changes to expose various properties that can be assigned for the selected observable.

**Figure 2.4: Manage Adjoint Observables Dialog Box**



Once the observable properties are defined ([Inputs for Observable Types \(p. 26\)](#)), click **Apply** to apply the settings and proceed to define other observables, or click **OK** in the **Manage Adjoint Observables** dialog box to apply the settings and close the dialog box. Available observables are described in [General Observables \(p. 6\)](#).

---

### Note

It is permissible to leave observable definitions incomplete. The **Manage Adjoint Observables** dialog box can be considered as a workspace for the definition and manipulation of observables. Only those operations with complete definitions will appear in the **Adjoint Observables** dialog box. Operations with undefined fields, or operations that depend on themselves, will be excluded from the list of usable observables.

## Inputs for Observable Types

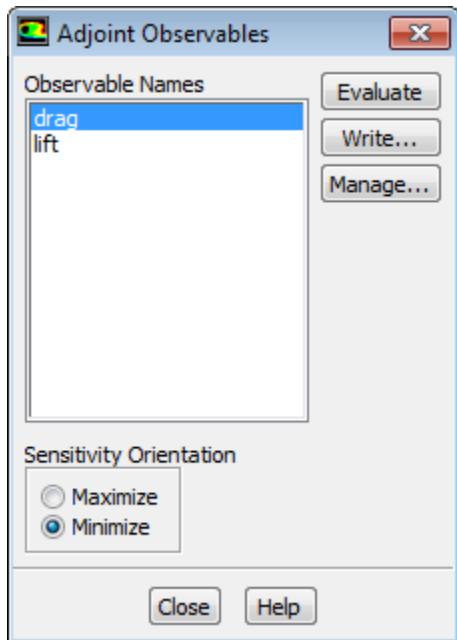
The inputs for each type of observable are summarized below:

- For a **force** observable:
  1. Select the walls that are to contribute to the force of interest in the **Wall Zones** list.
  2. Define the direction in which the force is to be computed by entering the components of this direction in the **X Component**, **Y Component**, and **Z Component** (for 3D) fields.
- For a **moment of force** observable:
  1. Select the **X**, **Y**, and **Z** (for 3D) components of the **Moment Center**.
  2. Select the **X**, **Y**, and **Z** (for 3D) components of the **Moment Axis**.
  3. Select a wall in the **Wall Zones** list.
- For a **swirl** observable:
  1. Select the **X**, **Y**, and **Z** (for 3D) components of the **Swirl Center**.
  2. Select the **X**, **Y**, and **Z** (for 3D) components of the **Swirl Axis**.
  3. Select a fluid in the **Fluid Zones** list.
- For a **pressure-drop** observable:
  1. Select the inlets and outlets between which the total pressure drop is to be computed using the **Inlets** and **Outlets** lists.
- For a **fixed value** observable, perform the following:
  1. Select the **Value** of the **Constant**.
- For a **surface-integral** observable:
  1. Select the type of integral from the **Integral Type** drop-down list. Available integral types are described in [General Observables \(p. 6\)](#).
  2. Select a **Surface** and the corresponding **Field Variable**.
- For a **volume-integral** observable:
  1. Select the type of integral from the **Volume Integral Type** drop-down list. Available integral types are described in [General Observables \(p. 6\)](#).
  2. Select the **Field Variable** of interest. If the chosen field variable is a vector quantity, you will need to specify a **Direction** along which the field variable will be evaluated.
  3. Under **Zones of Integration**, select whether to integrate over selected **Zones** or **Integrate over a box**.
  4. Select the **Zones** for integration under **Zone Settings** or the coordinates of the rectangular (in 2D) or hexahedral (in 3D) integration region under **Box Settings**.

- For a **ratio** observable:
  1. Select an observable from the **Numerator** drop-down list to represent the numerator of the ratio.
  2. Select an observable from the **Denominator** drop-down list to represent the denominator of the ratio.
- For a **product** observable:
  1. Select two observables from the corresponding drop-down list that will be used to compute their product.
- For a **linear combination** observable:
  1. Under **Linear Combination of powers**, select a **Constant** value.
  2. Select the number of **Components**.
  3. For each component, select a corresponding **Coefficient**, an **Observable**, and a **Power**.
- For an **arithmetic average** observable:
  1. Select the number of **Components**.
  2. Select observables from the corresponding **Observable** lists.
- For a **mean variance** observable:
  1. Select the number of **Components**.
  2. Select observables from the corresponding **Observable** lists.
- For a **unary operation** observable:
  1. Select an operation from the drop-down list (see [General Observables \(p. 6\)](#) for a list of available operators).
  2. Select an observable from the corresponding drop-down list upon which the unary operation is to be applied.

## 2.4.3. Selecting an Observable for Sensitivity Calculation

**Figure 2.5: Adjoint Observables Dialog Box**



You can select which observable to use for the sensitivity calculation in the list of **Observable Names**.

### Note

- Only one observable can be used in any one adjoint calculation. In the **Adjoint Observables** dialog box, the observable that is currently highlighted is the observable that will be used in the sensitivity calculation.
- Any observables that have been created but which are missing required inputs will not appear in the list of **Observable Names**.

Clicking the **Evaluate** button computes the current value of the selected observable quantity and prints the result in the console window. Clicking the **Write...** button provides the option to write the result to a named file.

The **Sensitivity Orientation** determines the sign of the post-processed sensitivities. For instance, if you select **Maximize** then changes in the direction of positive sensitivities will increase the value of the observable. A standard rule of thumb can be applied: if you want to improve the solution, then follow the direction of the sensitivity vectors. Another standard rule of thumb can also be applied for post-processed scalars: if you want to improve the solution, then increase values where the sensitivity is positive and/or decrease values where the sensitivity is negative.

## 2.5. Solving the Adjoint

Once an observable is defined as described in [Defining Observables \(p. 22\)](#), the solution of the adjoint for that observable can be computed based on the current flow solution. The steps for solving the adjoint are similar to those for solving the flow and are detailed in the following sections:

### 2.5.1. Using the Adjoint Solution Methods Dialog Box

### 2.5.2. Using the Adjoint Solution Controls Dialog Box

### 2.5.3. Working with Adjoint Residual Monitors

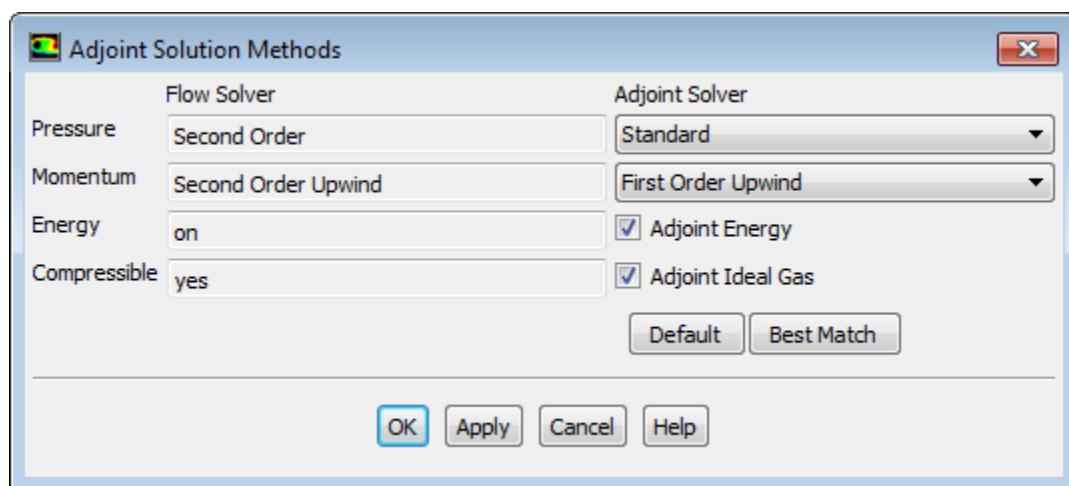
### 2.5.4. Running the Adjoint Calculation

## 2.5.1. Using the Adjoint Solution Methods Dialog Box

You can specify the methods used for computing the adjoint solutions in the **Adjoint Solution Methods** dialog box. To open the **Adjoint Solution Methods** dialog box, click **Methods...** in the **Design** ribbon tab (**Adjoint-Based** group box).

 Design → Adjoint-Based → Methods...

**Figure 2.6: Adjoint Solution Methods Dialog Box**



The **Adjoint Solution Methods** dialog box shows a side-by-side comparison of the schemes used for the flow solver and for the adjoint solver. Using the same scheme for the adjoint solution and the flow solution yields the most accurate discrete derivative calculation when the adjoint solution is converged. However, not all schemes used for the flow solver are supported for the adjoint solver. In these cases an alternate adjoint scheme must be used. This does not typically lead to severe deterioration of the adjoint results quality. Even if the same scheme is available for the adjoint solver this is not always practical because stability may be reduced with some schemes. Therefore, you can use the drop-down lists under **Adjoint Solver** to select alternate schemes as needed.

In some cases it may not be desirable to solve the energy adjoint even if energy is solved in the flow solver. For example, consider an incompressible flow where the specified observable does not involve thermal quantities. In this case the adjoint for the energy equation is identically zero, but its inclusion would add an unnecessary numerical burden. You can disable the **Adjoint Energy** option to avoid solving the energy adjoint.

If the primary flow is a compressible ideal gas, you can specify that the physics of the compressible flow are modeled in the adjoint solver by enabling the **Adjoint Ideal Gas** option. Enabling this option automatically enables the **Adjoint Energy** option as well. If the flow is not a compressible ideal gas, it will be treated as being incompressible for the purposes of the adjoint calculation.

Clicking **Default** will configure the adjoint solver to use the default low-order schemes (**Standard** for pressure and **First Order Upwind** for momentum), which are chosen for their stability. Clicking **Best Match** will attempt to match the adjoint schemes to the flow schemes that should in general provide

more accurate calculations at convergence. Where they cannot be matched, the default scheme will be used.

## 2.5.2. Using the Adjoint Solution Controls Dialog Box

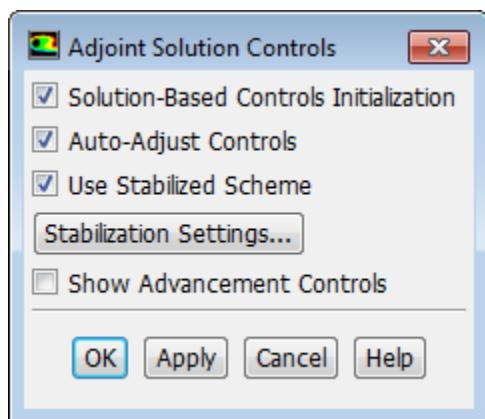
The advancement of the adjoint solution to reach a converged solution is an essential part of the analysis. The solution algorithm for the adjoint solver is similar to the coupled pressure-based solver that is available for conventional flow computations in ANSYS Fluent.

By default, the adjoint solver advancement settings will be automatically adjusted during calculation to encourage convergence, with initial values determined by the state of the flow solution. If needed, you can adjust the advancement controls in the **Adjoint Solution Controls** dialog box.

To open the **Adjoint Solution Controls** dialog box, click **Solver Controls...** in the **Design** ribbon tab (**Adjoint-Based** group box).

 **Design** → **Adjoint-Based** → **Solver Controls...**

### Options



#### Solution-Based Controls Initialization

When enabled, the advancement controls are selected automatically when the adjoint solution is initialized based on the state of the flow.

#### Auto-Adjust Controls

When enabled, the convergence of the AMG solver and the adjoint residuals are monitored during solution advancement. Based on the trends observed, the advancement controls are automatically adjusted to encourage reliable solution advancement and convergence. This is especially useful when a new type of problem is being solved for which appropriate settings are not initially clear. The advancement will begin with either user-specified settings or those from **Solution-Based Controls Initialization**. You can enable or disable **Auto-Adjust Controls** at any time during the solution.

If **Auto-Adjust Controls** is enabled, messages will appear as the calculation progresses indicating the adjustments that are being made.

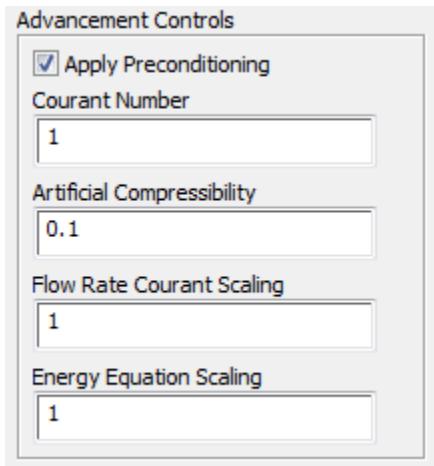
#### Use Stabilized Scheme

When enabled, stabilization will be applied to the adjoint solution. Click **Settings...** to specify the stabilization type and settings. The details of these settings are described in [Stabilized Scheme Settings \(p. 32\)](#). A stabilized solution scheme will be required to obtain adjoint solutions for problems at high Reynolds number in which there is strong shear and/or complex geometry.

## Show Advancement Controls

Toggles the visibility of the various advancement controls for direct user-specification.

## Advancement Controls



### Apply Preconditioning

Enables solution preconditioning. This is needed for most cases involving turbulent flow. If preconditioning is enabled, the following controls are available:

#### Courant Number

A higher number corresponds to a more aggressive advancement of the computation at the risk of instability. Disabling preconditioning corresponds to an infinite Courant number.

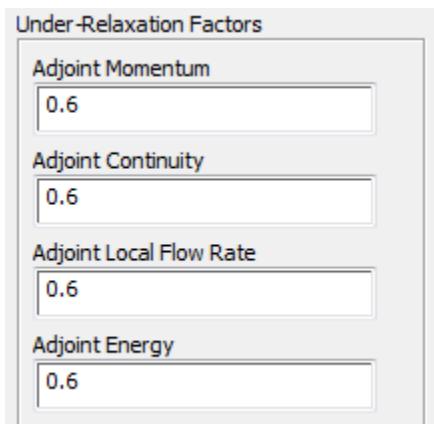
#### Artificial Compressibility

A nonzero value introduces artificial compressibility into the computation of the adjoint continuity equation. A value of 1.0 or less is reasonable.

#### Flow Rate Courant Scaling, Energy Equation Scaling

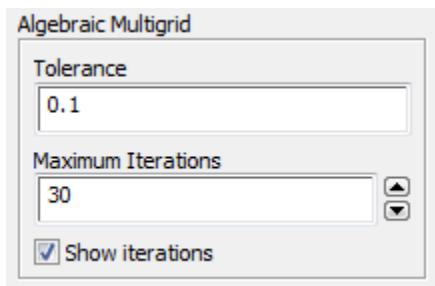
These values should be larger than zero (default values are 1). A smaller value implies a less aggressive algorithm that encourages stability of the AMG linear solver.

## Under-Relaxation Factors



**Adjoint Momentum, Adjoint Continuity, Adjoint Local Flow Rate, Adjoint Energy**

Each of these can be set to a value between 0 . 0 to 1 . 0. A higher value leads to a more aggressive algorithm that is less likely to be stable. A value of 1 . 0 for each can be used for some simple cases without difficulty.

**Algebraic Multigrid****Tolerance**

The tolerance used for judging convergence.

**Maximum Iterations**

The maximum number of inner iterations of the AMG solver.

**Show Iterations**

If enabled, a more verbose iteration history is printed in the text console during iterations. The details of the inner iteration can be useful when deciding on appropriate **Courant Number**, **Artificial Compressibility**, and **Flow Rate Courant Scaling**. If many inner iterations are needed, or if the inner iterations diverge, this signals that a reduction in **Courant Number** may be needed. Alternatively, an increase in **Artificial Compressibility** or a reduction in the **Flow Rate Courant Scaling** may be sufficient for the AMG iterations to converge.

### **2.5.2.1. Stabilized Scheme Settings**

If you choose to use one of the stabilization schemes ([Adjoint Solver Stabilization \(p. 15\)](#)) you will need to specify the scheme and settings in the **Stabilized Scheme Settings** dialog box (accessed by enabling **Use Stabilized Scheme** and clicking **Settings...** in the **Adjoint Solution Controls** dialog box).

The stabilization scheme settings are described in the following sections:

[2.5.2.1.1. Modal Stabilization Scheme](#)

[2.5.2.1.2. Spatial Stabilization Scheme](#)

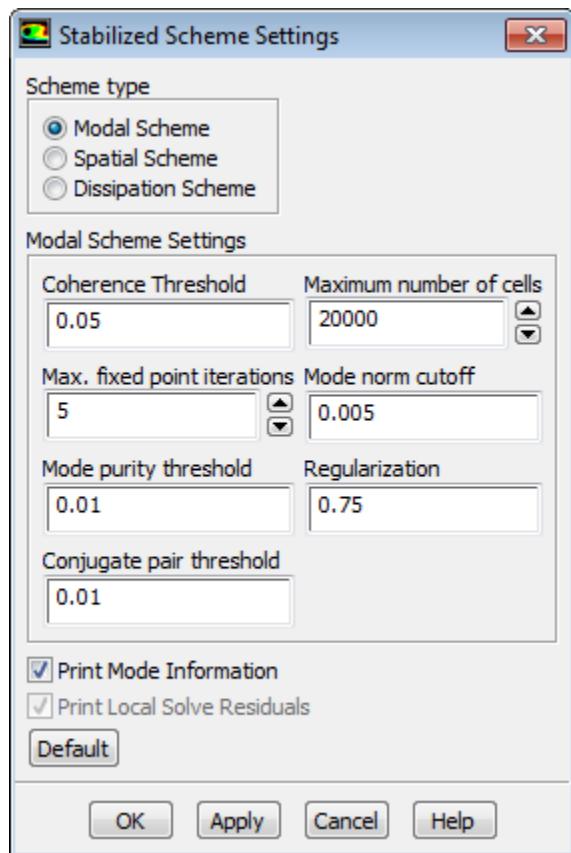
[2.5.2.1.3. Dissipation Scheme](#)

#### **2.5.2.1.1. Modal Stabilization Scheme**

The behavior of the modal scheme is as follows. As the calculation progresses the solution is monitored for unstable growing patterns or modes as the adjoint solution is corrected by the primary advancement algorithm. The first stage of the stabilization scheme involves a lightweight monitoring process to screen for any unstable patterns. If the calculation is progressing in a stable manner there is no indication even that the stabilization scheme is operating. If a candidate instability appears then it is monitored. In the event that unstable behavior that cannot be handled by the primary advancement scheme becomes evident a more intensive process is activated. Unstable patterns are refined using a fixed point iteration scheme so that they can be handled cleanly and advanced in a stable manner using an alternative to the primary advancement scheme.

The settings for the modal scheme are described below.

**Figure 2.7: Stabilized Scheme Settings for Modal Scheme**



### Coherence Threshold

controls the preliminary screening for growth. It defines the threshold for relative variance of growth between iterations at which a more detailed search is activated. A smaller value will tend to delay the activation of the more detailed search for unstable growth at the risk of allowing unstable growth to continue for more iterations.

### Max. Fixed Point Iterations

is the maximum number of iterations taken to refine the unstable pattern during the more detailed search.

### Mode Purity Threshold

is the tolerance for an acceptable unstable pattern. A smaller value will lead to a more pure unstable mode being identified. This may occur at the expense of a larger number of fixed point iterations. If the purity threshold is not met after the maximum specified number of iterations then the unstable pattern is rejected.

### Conjugate Pair Threshold

is the threshold for inclusion of a second unstable mode alongside a candidate mode, forming a conjugate pair. This corresponds to a pseudo-time-periodic instability, versus purely exponential growth. Such instabilities can occur in some cases. Increasing this value will increase the occurrence of a second mode being included.

### Maximum Number of Cells

is the largest number of cells allowed for an unstable pattern that will be handled by the stabilization scheme. The problematic instabilities for an adjoint solver are often localized in space to a small number

of cells out of the entire flow domain. However, there can be transiently growing corrections to the adjoint solution that masquerade as unstable patterns. This cell limit suppresses such false patterns from consideration.

### Mode Norm Cutoff

serves to identify the main body of an unstable pattern and exclude the peripheral spatial parts of the mode that are of small amplitude. If the norm cutoff were set to zero, all unstable modes would involve all cells in the domain. Therefore, reducing this value increases the number of cells that are included in an unstable mode.

### Regularization

acts to regularize the advancement of the unstable mode components. The stiffness of the adjoint governing equations can give rise to difficulty in discerning the unstable pattern from nearby stable patterns. Increasing the regularization tends to overcome this difficulty and adds stability to the calculation at the expense of potentially slower convergence. Valid values are between 0 and 1.

### Print Mode Information

enables printing of stabilization information to the console.

When using the modal stabilization scheme, the following behavior can be observed:

- When the primary advancement scheme is proceeding in a stable manner, the stabilization scheme will remain idle and no output will be generated.
- When a candidate unstable pattern appears it will be tracked and a message will appear with information about the detected mode:

```
Mode growth: 1.34390e+00 Coherence: 8.28546e-02
```

In the above example, the message indicates that an unstable pattern that grows by more than 34% in one iteration is present. The value of the Coherence corresponds to the variance in growth between iterations.

- When a candidate unstable pattern is identified that meets the coherence threshold, the secondary refinement stage is activated:

```
Mode growth: 1.35752e+00 Coherence: 3.95870e-02
Growing pattern cell count: 929
Isolating new mode
Iteration: 0 Growth rate: 1.42665e+00 Residual: 5.88196e-02
Iteration: 1 Growth rate: 1.41329e+00 Residual: 1.34600e-02
Iteration: 2 Growth rate: 1.41227e+00 Residual: 2.25348e-02
New growing mode with 597 cells
```

If the growing pattern cell count exceeds **Maximum Number of Cells** the refinement procedure is postponed. Likewise, if after refinement the number of cells involved exceeds the specified limit the mode is rejected. Each iteration corresponds to a refinement of the unstable pattern. The adjoint solution itself is not advanced during this process. The residual corresponds to the purity of the mode and must eventually fall below the **Mode Purity Threshold** for the mode to be accepted. A minimum of 3 iterations is taken so that the occurrence or absence of a conjugate pair can be verified.

- The presence of the sometimes violent unstable growth that can occur gives a jagged appearance to the adjoint residual plots. However, the underlying trend of the residual plots should correspond to a converging adjoint solution process.

The unstable patterns of solution advancement are specific to the flow solution, choice of discretization schemes, and solution advancement controls for the adjoint solver. If any of these values changes, then the unstable modal patterns must be recomputed so that the stabilization scheme will function to full effect. In the event that any of these parameters is changed and the adjoint solution is restarted a mode

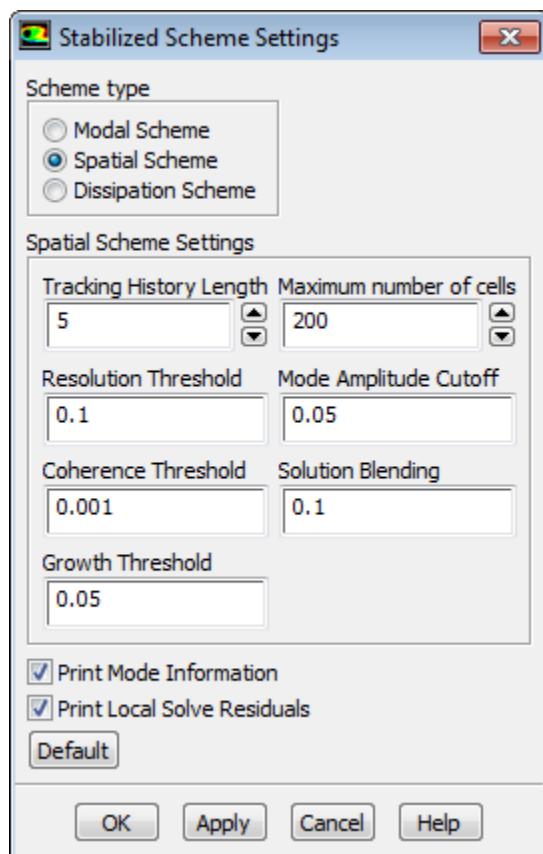
recalibration process will be initiated. It should be noted that if there are many unstable modes that have been identified then this recalibration may take some time.

In contrast, the patterns of unstable growth are not specific to the observable that has been selected. Given that there is computational effort needed to identify the instabilities, a separate initialization process is available just for the unstable modal patterns. This allows the adjoint solution itself to be initialized for a new observable while retaining the knowledge of how to stabilize the solution advancement. A much smoother path to solution for the second and subsequent observables is then achieved.

### 2.5.2.1.2. Spatial Stabilization Scheme

The spatial scheme operates by identifying parts of the domain where unstable growth is occurring and applying a more stable scheme. The unstable mode tracking and handling is done in an automated manner. Several settings are exposed, but are recommended for expert users only. When the stabilized scheme is used and the **Settings...** button is selected, the following dialog box will appear ([Figure 2.8: Stabilized Scheme Settings for Spatial Scheme \(p. 35\)](#)):

**Figure 2.8: Stabilized Scheme Settings for Spatial Scheme**



The four items on the left side of the dialog box relate to unstable mode tracking and selection, while the items on the right provide options for the response once a mode is identified. The values have the following meaning:

#### Tracking History Length

defines the number of iterations that are employed in the unstable mode identification process. A range between 2 and 5 may be chosen. The longer the tracking history, the better the quality of the identification process.

**Resolution Threshold**

defines a threshold for the separation between modes based on their amplitudes. A value of 0.1 implies that the second most-dominant mode must be 0.1 of the amplitude of the primary mode before an unstable mode can be considered as being adequately resolved.

**Coherence Threshold**

defines a threshold that must be met for the closeness of the mode to pure exponential growth as the iterations progress. A smaller number means that a tighter tolerance must be met before an unstable mode is considered as having been identified.

**Growth Threshold**

defines the growth rate threshold for unstable modes that must be met before the algorithm responds to the presence of the mode. The growth threshold defines the number of iterations that would have to take place for the mode to grow by one order of magnitude. That is, a value of 0.01 signifies that a mode that would grow one order of magnitude or more in 100 iterations will be handled. A negative value for this threshold is not recommended as multiple stable modes may be tracked.

**Maximum Number of Cells**

defines a limit on the number of cells for which stabilization will be applied when an unstable mode is located.

**Mode Amplitude Cutoff**

defines the fraction of the peak mode amplitude beyond which cells will be included for stabilization.

**Solution Blending**

defines an under-relaxation factor for blending the stabilized part of the scheme with the standard advancement scheme. A default of 0.1 is chosen. The value should lie between 0 and 1.

**Print Mode Information**

(when enabled) leads to details of the unstable mode tracking process being printed in the console window. The following is an example of what is printed when a mode is resolved partially:

```
Mode present> Growth : 2.38166e-01 Coherence : 1.27858e+00
Resolution Threshold : 1.00000e-01 Resolution : 1.32611e-01
```

The following is an example of what is printed when a mode becomes more fully resolved:

```
Mode present> Growth : 1.35318e-01 Coherence : 2.84133e-04
Resolution Threshold : 1.00000e-01 Resolution : 2.75312e-04
Growth Threshold : 5.60193e-02 Growth : 1.35318e-01
Coherence Threshold : 1.00000e-02 Coherence : 2.84133e-04
Local stabilization required: Instability detected in 5 cells
```

The final line only appears when a coherent and resolved instability has been identified successfully. Subsequently, the solution advancement algorithm automatically adjusts to eliminate this unstable behavior.

**Print Local Solve Residuals**

(when enabled) leads to details of the residuals for the local solution process being printed in the console window. An example of what is printed is given below:

Iter	Adjoint-continuity	Adjoint-xMom	Adjoint-yMom	Adjoint-Flowrate
...				
27	1.76334e-03	1.57503e-01	1.26984e-01	2.92618e-01
Local (%)	61.02095	57.41937	62.09185	99.99136

The percentage residual indicates the fraction of the total residual that is associated with the local solution process.

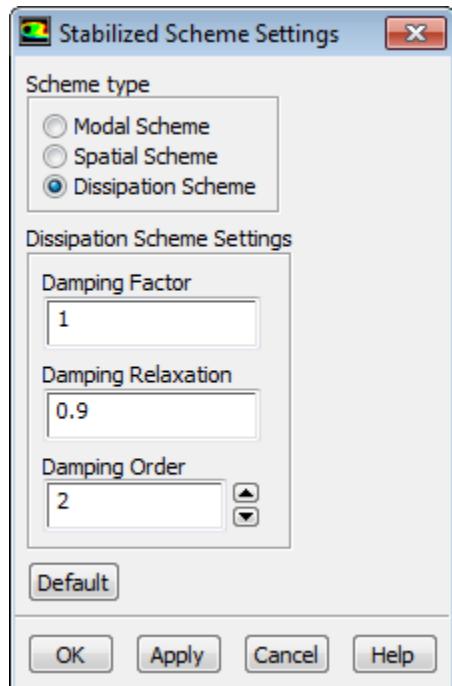
**Default**

provides a set of default values for the various fields in the dialog box.

**2.5.2.1.3. Dissipation Scheme**

The dissipation scheme operates by introducing nonlinear damping strategically into the calculation domain. The strategy is intended to provide minimal intervention in order to damp the growth of instabilities that lead to adjoint solution divergence. A marker is tracked, based on the state of the adjoint solution, and the damping is applied directly to the adjoint solution in regions where the marker becomes large.

**Figure 2.9: Stabilized Scheme Settings for Dissipation Scheme**



The following parameters can be set:

**Damping Factor**

The overall level of dissipation is proportional to this value although the damping level is determined ultimately through a nonlinear process.

**Damping Relaxation**

The damping relaxation can be used to control the rate at which the dissipation is updated as the adjoint solution progresses. A value of 1 freezes the dissipation, while a value of 0 means that the choice of dissipation is based purely on the adjoint solution's state at the previous iteration.

**Damping Order**

The spatial order of the dissipation. A higher order leads to more intense and localized damping. This can typically be set to be one order larger than the adjoint calculation spatial order.

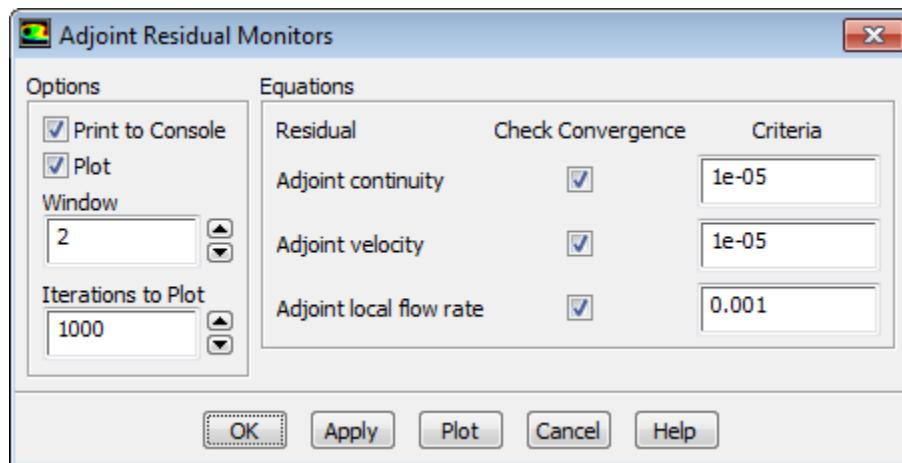
Since the order of discretization of the dissipation scheme is usually set higher than the adjoint, the formal order of accuracy of the adjoint solution is unaffected by the addition of dissipation. However, regions of intense dissipation can appear for some cases when using this scheme. This can sometimes appear as isolated spots and streaks in the adjoint solution on surfaces. The scale of these features is such that they are often easily smoothed during postprocessing to generate design changes.

### 2.5.3. Working with Adjoint Residual Monitors

The progress of the iterations of the adjoint solver and the monitoring of the convergence is controlled in the **Adjoint Residual Monitors** dialog box. This dialog box is accessed by clicking **Monitors...** in the **Design** ribbon tab (**Adjoint-Based** group box).



**Figure 2.10: Adjoint Residual Monitors Dialog Box**



The steps to define the monitor behavior are as follows:

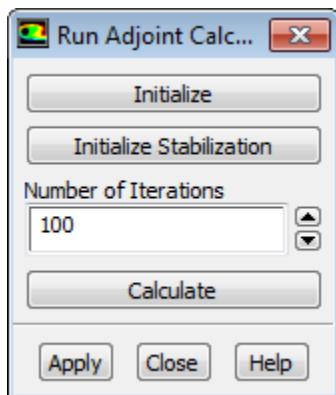
1. Decide whether or not the residuals are to be printed in the console by setting the **Print to Console** check box as desired.
2. Decide whether or not the residuals are to be plotted in the main window by setting the **Plot** check box as desired.
3. Set the id of the window in which the adjoint residuals are to appear in the **Window** field.
4. Set how many iterations are to be shown in the residual curves that are plotted in the **Iterations to Plot** field.
5. Enable and/or disable those values that are to be used as criteria for convergence, and set the **Criteria** in each case. The options are to test the residuals for the **Adjoint continuity** and/or **Adjoint velocity** and/or **Adjoint local flow rate** equations.

Click the **OK** or **Apply** buttons to confirm that the settings are acceptable. Clicking **Plot** will cause existing adjoint residuals to be plotted in the main graphics window.

### 2.5.4. Running the Adjoint Calculation

Initialization and execution of the adjoint solver is accomplished in the **Run Adjoint Calculation** dialog box. This dialog box is accessed by clicking **Calculate...** in the **Design** ribbon tab (**Adjoint-Based** group box).



**Figure 2.11: Run Adjoint Calculation Dialog Box**

The function of this dialog box is as follows:

#### **Initialize**

sets the value of the adjoint velocity and pressure to zero everywhere in the problem domain.

#### **Initialize Stabilization**

clears all of the unstable modes that have been identified in calculations using the modal stabilization scheme, but leaves the adjoint solution unchanged. Only available when **Modal** is selected in the **Stabilized Scheme Settings** dialog box.

#### **Calculate**

advances the adjoint solver by the **Number of Iterations** specified in the adjacent field.

Depending on the monitor settings, the residuals may be printed in the console and/or plotted in the main graphics window.

## 2.6. Postprocessing of Adjoint Solutions

Adjoint solution data can be post-processed so that it provides both qualitative and quantitative views of the effect of many types of change that may be imposed on a system. While shape changes are often of particular interest there is a very rich data set, one as large as the original flow field, available for exploration. Postprocessing tools are provided with the intention of permitting the adjoint data to be mined in such a way that it provides useful supporting information for an engineer who is making design decisions for a system, or has questions about the reliability of the original flow calculation.

Since the ANSYS Fluent adjoint solver is a discrete adjoint solver, the primitive adjoint solution data provides sensitivity to changes cell-wise for the computational mesh. Normalization of these results by the cell volume provides a *mesh-independent* view into the data set.

Information regarding postprocessing the adjoint solutions can be found in the following sections:

- [2.6.1. Field Data](#)
- [2.6.2. Scalar Data](#)

### 2.6.1. Field Data

Adjoint solution data can be post-processed using standard ANSYS Fluent postprocessing tools including contours, vectors, xy-plots, histograms, and surface and volume integrals.

In the **Contours** dialog box, under **Sensitivities...**, you can find the following fields:

### Magnitude of Sensitivity to Body Forces (Cell Values)

This field is the magnitude of the adjoint velocity primitive field. This field can be interpreted at the cell level as the sensitivity of the observable to a body force on the cell, or any factors that may change the momentum balance on the cell. As such it is a mesh-dependent quantity. This field is often observed to be large, for example, upstream of a body for which drag sensitivity is of interest, with the field diminishing in the upstream direction. This indicates the interference effect for an object positioned at various locations upstream of the object of interest.

### Sensitivity to Body Force X-Component (Cell Values), Sensitivity to Body Force Y-Component (Cell Values), and Sensitivity to Body Force Z-Component (Cell Values)

These are the components of the adjoint velocity primitive field. These fields show the sensitivity of the observable to the various components of a body force that may be applied to a cell, or other factors that may change the momentum balance on the cell. As such, it is a mesh-dependent quantity.

### Sensitivity to Mass Sources (Cell Values)

This field provides a plot of the primitive adjoint pressure field. This field can be interpreted at the cell level as the sensitivity of the observable with respect to any changes to the mass balance on the cell. As such, it is a mesh-dependent quantity.

### Sensitivity to Energy Sources (Cell Values)

This field is available when the energy adjoint is solved and is the primitive adjoint temperature field. It can be interpreted at the cell level as the sensitivity of the observable with respect to changes in the thermal energy balance of the cell. As such, it is a mesh-dependent quantity.

### Magnitude of Sensitivity to Body Forces

This field is the magnitude of the adjoint velocity primitive field normalized by cell volume. This field can be interpreted as the magnitude of the sensitivity of the observable to body force per unit volume. It can be used to identify regions in the domain where small changes to the momentum balance in the flow can have a large or small effect on the observable. This field is often observed to be large, for example, upstream of a body for which drag sensitivity is of interest, with the field diminishing in the upstream direction. This indicates the interference effect for an object positioned at various locations upstream of the object of interest. This field is normalized so that, in principle, it shows a mesh-independent quantity. In practice, this field has features that are often correlated with mesh structure. This is not unexpected. In fact, the adjoint solution field is providing very useful insight into regions of the flow domain where the observation of interest is potentially sensitive to particular features of the mesh and discretization process that affect the local momentum balance.

### Sensitivity to Body Force X-Component, Sensitivity to Body Force Y-Component, and Sensitivity to Body Force Z-Component (in 3D) fields

These are the components of the adjoint velocity primitive field normalized by cell volume. This field can be interpreted as the magnitude of the sensitivity of the observable to components of a body force per unit volume. Consider a body force distribution, expressed as a force per unit volume. The integral of the vector product of that distribution with the components of this field gives a first-order estimate of the net effect of the body force on the observation.

### Sensitivity to Mass Sources

This field is the primitive adjoint pressure field normalized by cell volume. This field can be interpreted as the sensitivity of the observable with respect to mass sources or sinks in the domain. Consider a mass source/sink distribution, expressed as mass flow rate per unit volume. The integral of that distribution, weighted by the local value of this field, gives the effect of the sources/sinks on the observation. This field is normalized so that, in principle, it shows a mesh-independent quantity. In practice, this field has features that are often correlated with mesh structure. This is not unexpected. In fact the adjoint solution field is providing very useful insight into regions of the flow domain where the obser-

vation of interest is potentially sensitive to particular features of the mesh and discretization process that affect local mass balance. When plotted on a boundary, this field indicates the effect of the addition or removal of fluid from the domain upon the quantity of interest. It is important to note that in this scenario the effect of the momentum of the fluid that is added or removed is not taken into account. The boundary velocity sensitivity should be plotted if that effect is also of interest.

### Sensitivity to Energy Sources

This field is available when the energy adjoint is solved and is the primitive adjoint temperature field normalized by cell volume. It can be interpreted at the cell level as the sensitivity of the observable with respect to thermal energy sources or sinks in the domain. This field is normalized so that, in principle, it is a mesh-independent quantity.

### Sensitivity to Viscosity

This field shows the sensitivity of the quantity of interest to variations in the turbulent effective viscosity for a turbulent problem, or the laminar viscosity in a laminar case. The sensitivity is normalized by the cell volume to account for cell size variations in the mesh.

### Shape Sensitivity Magnitude

This field is the magnitude of the sensitivity of the observable with respect to a deformation applied to the mesh (both boundary and interior mesh). When plotted on the surface of a body the locations where this quantity is large indicates where small changes to the surface shape can have a large effect on the observable of interest. If the shape sensitivity magnitude is small then the effect of shape changes in this region can be expected to have a small effect on the observable of interest. When viewing this field, it is often observed that the magnitude varies by many orders of magnitude. Contour plots will clearly draw attention to regions with the highest sensitivity (often sharp edges and corners). However, it should be remembered that a relatively small surface movement that is distributed over a large area can have a cumulative effect that is large.

### Normal Shape Sensitivity

This field shows the normal component of the shape sensitivity. A positive value indicates an orientation directed into the domain, while a negative value indicates that the shape sensitivity is oriented outwards from the domain. This field eliminates the component of the vector shape sensitivity field that lies in the plane of the wall.

### Normal Optimal Displacement

This field shows the normal component of the optimal displacement computed from the adjoint solution. This field is defined only for portions of walls lying within the control-volume specified for morphing. The size of the optimal displacement is determined by the scale factor that is chosen for the morphing. A positive value of displacement indicates that the surface will be displaced into the flow domain, whereas a negative value of displacement corresponds to wall movement outwards from the flow domain. This field eliminates the component of the optimal displacement vector that lies in the plane of the wall.

### **log10(Shape Sensitivity Magnitude)**

In view of the large range of values possible for the shape sensitivity magnitude, a convenience function that plots  $\log_{10}$  of the magnitude is provided. This allows the importance of the surfaces in a domain to be ranked more easily based on how they affect the observation of interest when they are reshaped.

### **Shape Sensitivity X-Component, Shape Sensitivity Y-Component, and Shape Sensitivity Z-Component (in 3D) fields**

These fields are the individual components of the sensitivity of the observable of interest with respect to the mesh node locations. It is plotted as cell data and is computed as the average of the nodal

sensitivities for a given cell, divided by the cell volume. Note that for this discrete adjoint solver the sensitivity of the result with respect to node locations both on and off boundaries is computed. The normalization by cell volume indicates that the fields that are plotted are the weighting factors for a continuous spatial deformation field. (Note that the nodal sensitivity data itself is used when mesh morphing is performed, and predictions about the effect of shape changes are made.)

### **Sensitivity to Boundary X-Velocity, Sensitivity to Boundary Y-Velocity, and Sensitivity to Boundary Z-Velocity (in 3D) fields**

These fields are defined on those boundaries where a user-specification of a boundary velocity is made for the original flow calculation. This includes no-slip walls. The field shows how sensitive the observable of interest is to changes in the boundary velocity at any point. It is interesting to note that even though the original boundary condition specification may be for a uniform velocity on the domain boundary, the effect of a non-uniform velocity perturbation is available. The effect of any specific boundary velocity change can be estimated as an integral of the vector product of the change to the velocity with the plotted sensitivity field. A plot of this quantity on a velocity inlet, for example, can be very useful for assessing whether or not the inlet is positioned too close to key parts of the system. That is, it addresses the question of whether or not the flow domain is too small to achieve a successful computation of the performance measure of interest. Viewing this field will also indicate whether or not the assumption of a uniform inflow is adequate.

### **Sensitivity to Boundary Pressure**

This field is defined on boundaries where there is a user-specified pressure as part of a boundary condition, such as on a pressure outlet. The field shows the sensitivity of the observation of interest to variations in the boundary pressure across the flow boundary. It is interesting to note that even though the original boundary condition specification may be for a uniform pressure on the domain boundary, the effect of a non-uniform pressure perturbation is available. The effect of any specific boundary pressure change can be estimated as an integral of the product of the change to the pressure with the plotted sensitivity field. Viewing this field will also indicate whether or not the assumption of a uniform pressure is adequate for the simulation.

### **Sensitivity to Boundary Temperature**

This field is available when the energy adjoint equation is solved and is defined on boundaries where a temperature boundary condition is applied. This includes walls, velocity inlets, and pressure outlets where a backflow temperature may be specified. The field shows the sensitivity of the observation of interest to variations in the boundary temperature across the boundaries. Note that even if the original boundary condition specification is for a uniform temperature on the boundary, the effect of a non-uniform temperature perturbation is available. The effect of any specific boundary temperature change can be estimated as an integral of the product of the change to the temperature with the plotted sensitivity field. This field can be used to indicate whether or not the assumption of a uniform temperature is adequate for the simulation.

### **Sensitivity to Boundary Heat Flux**

This field is available when the adjoint energy equation is solved and is defined on walls where a heat flux boundary condition is imposed. The field shows the sensitivity of the observation of interest to variations in the boundary heat flux through the wall. Its properties are analogous to those of Sensitivity to Boundary Temperature.

### **Sensitivity to Flow Blockage**

This field is provided as a convenient tool for identifying portions of the flow domain where the introduction of blockages or obstructions in the flow can affect the observation of interest. Consider a blockage in the flow that generates a reaction force on the flow that is proportional to the local flow speed, and acting in the opposite direction to the local flow:  $F = -\alpha(r)v$  where  $\alpha$  is a local coefficient for the reaction force. The local contribution of this force on the observation of interest is determined

by the vector product of this force with the adjoint velocity field. The flow blockage field that is plotted is  $-\boldsymbol{v} \cdot \tilde{\boldsymbol{v}}$ , namely the negative of the vector product of the flow velocity and the adjoint velocity (Cell Value).

### **Adjoint Local Solution Marker**

This field, intended for expert users, can be plotted to identify those portions of the flow domain where the stabilized adjoint solution advancement scheme is applied. It is preferable to plot this with the **Node Values** de-selected in the **Contours** dialog box. In this case, the **Adjoint Local Solution Marker** will take a value between 0 and 1. The **Mode Amplitude Cutoff** defined in the **Stabilized Scheme Settings** dialog box defines the lower bound for cells where the stabilized scheme is applied.

The surface shape-sensitivity fields contain the derivatives of the chosen observable with respect to the position of the bounding walls of the problem.

In the **Vectors** dialog box, under **Vectors of...**, you can find the following custom vector fields:

### **Sensitivity to Body Forces (Cell Values)**

This vector field shows how sensitive the observable of interest is to the presence of body forces within the flow. A body force that locally has a component in the direction of the body-force sensitivity vector will lead to an increase in the observation of interest. The larger the sensitivity vector, the larger the effect that a body force can have on the observation of interest.

### **Sensitivity to Body Forces**

This vector field shows how sensitive the observable of interest is to the presence of body forces within the flow. It is normalized by the local cell volume so that it is *mesh-independent* data. A body force that locally has a component in the direction of the body-force sensitivity vector will lead to an increase in the observation of interest. The larger the sensitivity vector the larger the effect that a body force can have on the observation of interest. The first-order effect of a body force distribution per unit volume can be computed as a volume integral of the vector product of the body force distribution with this field.

### **Sensitivity to Shape field defined on walls**

This vector field shows the post-processed adjoint solution, independent of any morphing settings, that defines the sensitivity of the observable of interest to movement of the walls. Deformation of a wall where this vector is large can be expected to have a significant effect on the observable. In contrast, deformations of the wall in locations where this vector is small in magnitude, to first order, will have no effect on the observable.

### **Optimal Displacement**

The optimal displacement field is defined once a design change has been computed using the design tool. This vector field shows the optimal displacement based on the design goals that have been specified and any prescribed boundary motions. Note that when plotting this vector field, if the **Auto Scale** option is deselected, and the vector **Scale** option is set to 1, then the vectors as shown display the absolute displacement that will occur.

### **Sensitivity to Boundary Velocity**

This sensitivity field is defined on boundaries where velocities are specified as an input for the boundary condition. This includes no-slip, and slip walls. When plotted, this field illustrates where a change to the boundary velocity can affect the observation of interest. When plotted on a wall, it shows how the imposition of a nonzero velocity condition on the wall will affect the observation. This can include movement in the plane of the wall itself. When plotted, for example on a velocity inlet, this field illustrates how local adjustments to the velocity of the incoming flow affect the observation of interest. Integrating a potential change to the velocity field, weighted by this sensitivity field over the boundary provides a first order estimate of the effect of the change on the observation of interest.

### Sensitivity to Boundary Mass Flux

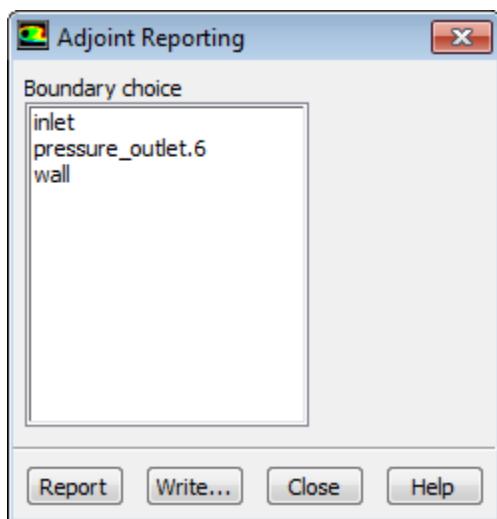
This sensitivity field that can be used to visualize the effect of adding or removing fluid via a wall-normal jet. This can be used, for example, to identify regions on walls where the introduction of suction could have a significant effect on the observable that has been specified. A plot of the **Sensitivity to Boundary Velocity** in general provides a richer view, but sometimes at the expense of the clarity of the visualization.

### 2.6.2. Scalar Data

The computed adjoint solution can provide sensitivity information for individual boundary condition settings through the **Adjoint Reporting** dialog box. This dialog box is accessed by clicking **Reporting...** in the **Design** ribbon tab (**Adjoint-Based** group box).



**Figure 2.12: Adjoint Reporting Dialog Box**



Boundary condition sensitivity data is retrieved as follows:

- Select a single boundary of the problem in the **Boundary choice** selection box.
- Click **Report** to see the boundary condition settings and the sensitivity of the observable with respect to those settings. An example of a report is shown below:

```
Velocity inlet id 5
Velocity magnitude = 4.000000000e+01 (m/s),
Sensitivity = 5.429127866e+01 ((n)/(m/s))
```

In this case, the report shows that for every change of the boundary velocity by 1 m/s, a change in the predicted observable (in this case a force) of 54.2 N is expected. This prediction is based on a linear extrapolation.

- Click **Write...** to write the report for the selected boundary to a file.

### 2.7. Modifying the Geometry Using the Design Tool

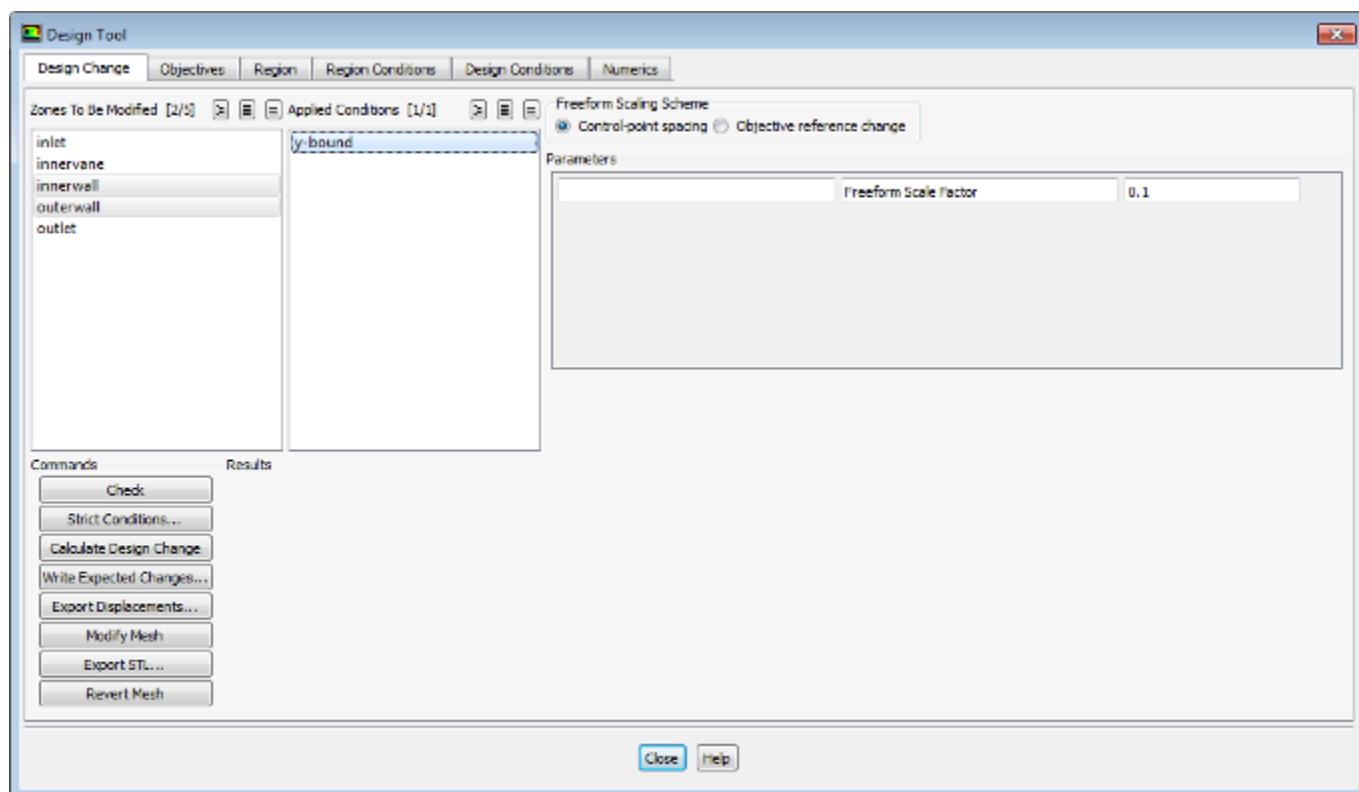
Tools to modify the boundary and interior mesh of the problem based on the adjoint solution are provided in the **Design Tool** dialog box. The **Design Tool** uses computed adjoint sensitivity data to find an optimal design change satisfying one or more goals or constraints, and to predict the resulting changes in observable values. Among the goals that can be specified in the **Design Tool** are:

- minimization, maximization, or targeted changes of single or multiple observable values
- prescribed deformations of selected geometry
- general constraints on deformations including restrictions on direction of motion, symmetry, and boundary conditions.
- equality and/or inequality constraints on geometry (fixed walls, bounding boxes, and so on).

The **Design Tool** is accessed by clicking **Design Tool...** in the **Design** ribbon tab (**Adjoint-Based** group box).



**Figure 2.13: Design Tool Dialog Box**



The general procedure to use the Design Tool is as follows.

- Select those boundaries that are free to be deformed in the **Zones To Be Modified** field on the **Design Change** tab. Boundaries that are not selected in this box that intersect the control volume will be constrained not to move. This is useful if there are walls which, for design reasons, must not move.
- Define the region that will be modified in the **Region** tab. ([Defining the Region for the Design Change \(p. 46\)](#))
- Define conditions on the deformation region such as number of control points, symmetry, directional invariance, and boundary continuity. ([Defining Region Conditions \(p. 47\)](#))
- Compute and export the sensitivities from the adjoint solution for each observable of interest. ([Exporting Sensitivity Data \(p. 47\)](#))

5. Import the sensitivity data for each observable that will participate in the design change and specify the optimization goals for the observables. ([Defining Observable Objectives \(p. 48\)](#))
- 

**Note**

Export and Import of sensitivity data is only required for multi-objective analysis.

---

6. Define constraining and/or deformation conditions on the design change, such as fixed boundaries, prescribed boundary deformation (through a profile, translation, rotation, or scaling), and bounding surfaces / planes. ([Defining Conditions for the Deformation \(p. 49\)](#))
7. Specify weights for the various objectives, a scaling factor for the deformation, and whether the design conditions are strictly enforced. ([Shape Modification \(p. 53\)](#))
8. (optional) Adjust the optimal design calculation numerics if required.
9. Compute the optimal design change and modify the mesh.

Details of these steps are elaborated in the following sections:

[2.7.1. Defining the Region for the Design Change](#)

[2.7.2. Defining Region Conditions](#)

[2.7.3. Exporting Sensitivity Data](#)

[2.7.4. Defining Observable Objectives](#)

[2.7.5. Defining Conditions for the Deformation](#)

[2.7.6. Shape Modification](#)

[2.7.7. Design Tool Numerics](#)

## 2.7.1. Defining the Region for the Design Change

The following procedure can be used to define the design change region:

1. Click the **Region** tab in the **Design Tool** dialog box.
  2. Enable the **Show Bounding Box** option to make a bounding box visible when the mesh is displayed.
  3. Click the **Get Bounds...** button to open the **Bounding Box Definition** dialog box. Select the **Zones to be bounded** from the list of zones. This will populate the **X Min**, **X Max**, **Y Min**, and **Y Max** (and **Z Min** and **Z Max** in 3D) fields with the bounding box for the selected zones. Click the **Larger Box** and **Smaller Box** buttons to adjust the control volume size as needed. The limits of the bounding box can be entered individually by hand if desired.
- 

**Note**

If the deformation region includes symmetry planes that are to be respected in the deformation, the control volume should be defined such that it is bisected by the symmetry planes. The symmetry planes must be oriented with normals in the Cartesian coordinate directions.

---

4. Click **Update Region**. This will set the values for the region boundaries and update the display of the design change region.

## 2.7.2. Defining Region Conditions

1. Click the **Region Conditions** tab in the **Design Tool** dialog box.
2. Specify the control points and motion conditions for each coordinate direction.
  - a. Select the number of control **Points**.  
The control points are distributed uniformly in each coordinate direction. The spacing between the control points defines the characteristic spatial scale for the morphing operation. The larger the number of control points, the smaller the spacing and hence, the smaller the spatial scale on which changes can be made. 20 to 40 control points for each coordinate direction is typical.
  - b. If you want to prevent deformation of the control volume in the given coordinate direction, un-check **Motion Enabled**.
  - c. If you want the motion along a coordinate direction to be uniform in one or more directions, enable **Invariant in X**, **Invariant in Y**, and **Invariant in Z** (3D) as appropriate.
  - d. If the deformation region includes a symmetry boundary with a normal in one of the Cartesian coordinate directions, you can enable **Symmetric in X/Y/Z** as appropriate to allow morphing in the symmetry plane.

### Note

The deformation region must be defined such that it is bisected by the symmetry plane(s).

3. The **Region Boundary Continuity** settings control the order of continuity of the morphing process at the boundaries of the control volume. This is of particular relevance when the boundary of the control volume intersects a wall. The **Continuity Order** controls the smoothness of the transition from the unmoved wall lying outside the control volume to the deformed wall within. A value of 1 ensures first order continuity at the transition, in other words, a continuous first derivative of the morphing operation. A value of 2 ensures second order continuity, and so forth.

By default, the continuity **Definition** is **Uniform** and is enforced at all control volume boundaries. Alternatively, you can choose **By Boundary** to use different continuity conditions at each control-volume boundary. For each boundary (**X-Min**, **X-Max**, etc.) you can specify what order continuity to enforce for each motion direction, as well as specify **In-Plane Motion Only** on that boundary.

The continuity settings will not affect the transition between fixed and movable boundaries within the control volume, only mesh nodes in the transition zone from inside to outside the control volume.

4. Click **Apply**.

## 2.7.3. Exporting Sensitivity Data

In order to perform multi-objective optimization, you need to export the sensitivity data for each observable/flow-condition combination that will take part in the optimization.

1. Define the observables of interest as described in [Defining Observables \(p. 22\)](#).

2. For each observable solve for and export the sensitivity data.
  - a. Solve the adjoint as described in [Solving the Adjoint \(p. 28\)](#).
  - b. In the **Region** tab of the **Design Tool** dialog box, specify the deformation region and click **Export Sensitivities...**

The exported file contains the sensitivities of the observable for the mesh nodes within the specified deformation region. The deformation regions for the various observables can, in principle, be different. However, it is advisable that there is as much overlap as possible for observables that will be used together in a multi-objective design process.

## 2.7.4. Defining Observable Objectives

The Design Tool computes the optimal mesh deformation to satisfy multiple simultaneous design goals. Among these can be goals for multiple observables, possibly at multiple flow conditions. Once you have obtained the adjoint solutions and displacement sensitivities for the various observables of interest, you can import the data and specify the design objectives for each.

1. Click the **Objectives** tab in the **Design Tool** dialog box.
2. To import previously saved sensitivity data, click **Manage Data...**. In the **Manage Sensitivity Data** dialog box click **Import Sensitivities...** to load previously saved sensitivity files and close the **Manage Sensitivity Data** dialog box.
3. If you want to include computed sensitivity data in the current session that you have not exported yet enable **Include current data**.
4. For each observable objective you can specify the following.

### Objective

determines how Fluent will attempt to change the value of the observable through the design change. You can choose to increase or decrease the value of the observable optimally, or you can specify a target change in value. You can also select **None** in which case the observable will not be considered in computing the optimal design change, but the predicted change in observable value will still be reported.

### Target/Reference Change

When the chosen **Objective** is **Target Change in Value**, this field corresponds to the desired change in the value. When the chosen **Objective** is **Maximize** or **Minimize**, the magnitude of this field is a reference weighting for the observable. This can be used to normalize the objectives in cases where you have multiple observables of vastly differing magnitudes, thereby allowing the values entered for **Weight** in the **Design Change** tab to all be of similar scale. You can check **As Percentage** to specify the **Target/Reference Change** as a percentage of the value of the observable.

---

### Note

You must click **Apply** for each objective you define before moving on to the next one.

---

## 2.7.5. Defining Conditions for the Deformation

In the **Design Conditions** tab, you can specify conditions on the mesh morphing such as fixing portions of wall surfaces, prescribing the deformation of surfaces, or setting up bounding planes / surfaces. To create a condition, click **Create...**, select the type of condition in the **Create New Condition** dialog box (you can choose between **fixed-walls**, **bounded-by-plane**, **bounded-by-surfaces**, **prescribed-profile**, **rotation**, **translation**, and **scaling**), and provide a unique **Name**. You can then define the condition by selecting it in the list on the left, defining the settings (as described in the steps that follow), and clicking **Apply**. If necessary, you can **Rename** a selected condition using the **Rename Constraint** dialog box that opens. Once created, the design conditions can be individually included or excluded from the design change in the **Design Change** tab.

### Important

Note that a given surface cannot have multiple conditions of type **prescribed-profile**, **rotation**, **translation**, or **scaling**.

### Fixed Walls

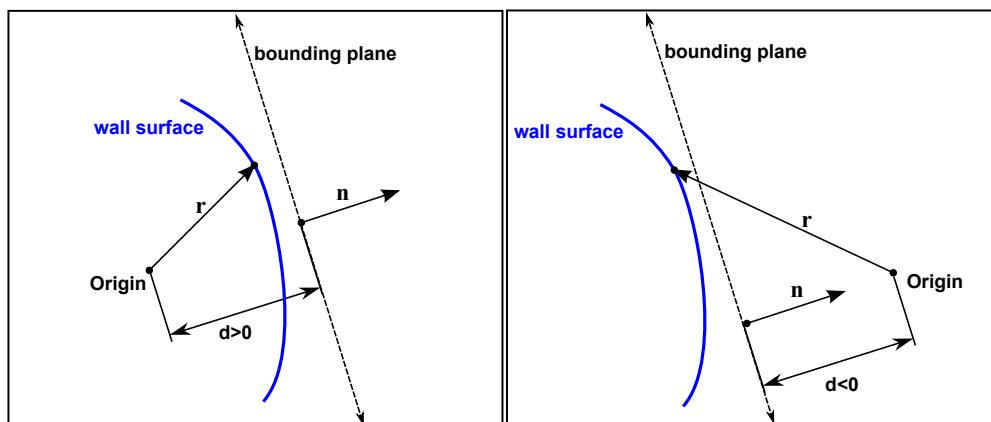
The **fixed-walls** constraint can be applied to clip-surfaces of walls in the deformation region. This offers a fine-grained control over parts of the geometry that should not move. Whereas deselecting a zone in the **Zones To Be Modified** list on the **Design Change** tab will fix an entire wall zone, selecting a clip-surface from in the **Fixed Walls Condition** list will fix only the portion of the wall that forms the clip-surface.

Any clip surfaces you have defined will appear in the **Clip Surfaces** list. You can assign clip surfaces to a fixed walls constraint by selecting them in the list and clicking **Apply**.

### Bounding Planes

The **bounded-by-plane** condition allows you to define one or more bounding planes that constrain the geometry deformation in such a way that the geometry cannot cross the given planes, thereby respecting packaging or form-factor constraints. The bounding plane is characterized by a vector normal to the plane and a distance measured from the global coordinate system origin to the plane, along the normal direction. The motion of the selected wall surface(s) will be constrained such that all points on the surface(s) lie on the side of the bounding plane that is opposite to the normal direction.

**Figure 2.14: Specifying a Bounding Plane for Design Changes**



In [Figure 2.14: Specifying a Bounding Plane for Design Changes \(p. 49\)](#) the bounding plane normal vector is denoted by  $\mathbf{n}$  and the distance is denoted by  $d$ .  $\mathbf{r}$  is the position vector of a point on the wall surface subject to the bounded-by-plane condition. Note that a positive value for the distance yields a bounding plane that is oriented with its normal pointing away from the origin, while a negative value for the distance yields a bounding plane oriented with its normal direction pointing towards the origin. The constraint equation can be expressed as:

$$\mathbf{r} \cdot \frac{\mathbf{n}}{|\mathbf{n}|} \leq d$$

1. Select the surfaces (or clip-surfaces) that are subject to the bounding plane condition in the **Bounded Surfaces** list.
2. Enter the components of the bounding plane normal direction.
3. Enter the distance of the plane from the origin (along the normal direction).
4. Click **Apply**.

## Bounding Surfaces

The **bounded-by-surfaces** condition allows you to import one or more bounding surfaces that constrain the geometry deformation in such a way that the geometry cannot cross the nearest facet of the bounding surfaces, thereby respecting packaging or form-factor constraints. The bounding surfaces can be read from an .stl, .msh, or .cas file. The deformation of the selected bounded surfaces will be constrained such that all points on the surfaces lie on the sides of the bounding surfaces that have a positive orientation.

1. Select the surfaces (or clip-surfaces) that are subject to the bounded-by-surface condition in the **Bounded Surfaces** list.
2. Use the **Read...** button to import a .stl, .msh, or .cas file that contains the appropriate bounding surface(s). Note that the file you read must have the same dimensionality (2D or 3D) as the working case file.
3. Select the **Imported Surfaces** that will constrain the bounded surfaces.
4. To visually confirm that the selected **Bounded Surfaces** and **Imported Surfaces** are appropriate, click the **Display** button. The imported surfaces will be colored green by default.

---

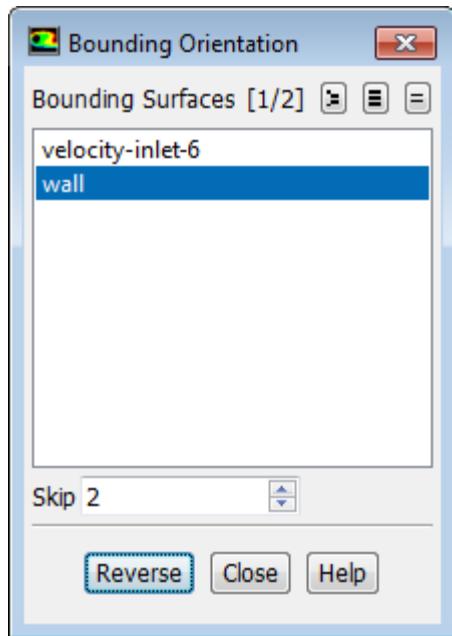
### Note

When examining the surfaces, it may be helpful to revise display properties of particular surfaces (for example, the color, the level of transparency, and the visibility of the mesh edges or faces). This can be done using the **Scene Description** dialog box, as described in [Composing a Scene in the \*Fluent User's Guide\*](#). Note that you can even revise the display properties of the imported surfaces, the names of which will be displayed with the prefix **bounding-**.

- 
5. To ensure that the **Bounded Surfaces** are constrained on the correct side of the **Imported Surfaces** (that is, the side that has a positive orientation), click the **Orientation...** button. The selected **Imported Surfaces** will be displayed, with their orientation indicated by arrows. In the dialog box that opens ([Figure 2.15: The](#)

[Bounding Orientation Dialog Box \(p. 51\)](#)), you can select individual **Bounding Surfaces** and **Reverse** their orientation as necessary; you can also increase the **Skip** value to display less orientation arrows, for cases where the arrow density obscures the orientation. Note that the orientation you define for a particular bounding surface will be used globally for all **bounded-by-surfaces** conditions.

**Figure 2.15: The Bounding Orientation Dialog Box**



When reviewing the orientation of a bounding surface, you should verify that the facets have a uniform orientation, that is, all of the arrows are on the same side of the surface. Bounding surfaces with facets that have a mixed orientation can affect the convergence and/or accuracy of the calculation.

6. Click **Apply** in the **Design Conditions** tab of the **Design Tool** dialog box to save your settings.

---

### Note

When using the **bounded-by-surfaces** condition, it is recommended that you apply preconditioning on the advancement of the process for updating the freeform displacements (by changing the **Preconditioning** field to a positive value in the **Numerics** tab of the **Design Tool** dialog box, as described in [Design Tool Numerics \(p. 56\)](#)), in order to encourage the stability of the solution.

---

## Prescribed Profiles

You can use the **prescribed-profile** condition to assign a motion profile to a wall zone or zones using a **DEFINE\_GRID\_MOTION** UDF.

1. Select the wall zones for which to prescribe the displacement in the **Wall Zones** list.
2. Compile your **DEFINE\_GRID\_MOTION** UDF in Fluent. (Refer to [DEFINE\\_GRID\\_MOTION in the Fluent Customization Manual](#)).
3. Select your UDF in the **Deformation profile** drop-down list.

4. Specify the **Scale Factor** to be applied to the deformation profile. Alternatively, you can specify or create an input parameter to use for the **Scale Factor** by clicking the **P** button next to the **Scale Factor** field.
5. Click **Apply**.

## Rotation

The **rotation** condition allows you to specify that surfaces undergo rotation.

1. Select the surfaces (or clip-surfaces) you want to rotate in the **Surfaces** list.
2. Enter the rotation **Angle** in degrees. Note that you can enter a set value, or use the **P** button to assign an input parameter (for example, as part of a parametric study managed by Workbench); see [Select Input Parameter Dialog Box in the \*Fluent User's Guide\*](#) for details.
3. Define the **Origin** about which you want the surfaces to rotate.
4. For 3D cases, define the **Axis** of rotation. For 2D cases, this is automatically defined as the positive Z axis.
5. Click **Apply**.

## Translation

The **translation** condition allows you to specify that surfaces undergo translation.

1. Select the surfaces (or clip-surfaces) you want to translate in the **Surfaces** list.
2. Define the **Displacement** to be applied along each coordinate axis. Note that for each field you can enter a set value, or use the **P** button to assign an input parameter (for example, as part of a parametric study managed by Workbench); see [Select Input Parameter Dialog Box in the \*Fluent User's Guide\*](#) for details.
3. Click **Apply**.

## Scaling

The **scaling** condition allows you to specify that surfaces are scaled along one or more axes.

1. Select the surfaces (or clip-surfaces) you want to scale in the **Surfaces** list.
2. Select the scaling **Type** and define the associated settings:

- **Radial**

This type scales the surfaces radially about the **Origin**, using a single scaling **Factor** along each of the coordinate axes. The size of the surfaces will change, though not the shape.

- **In-Plane**

This type is available for 3D cases, and scales the surfaces about the **Origin** using a single scaling **Factor** along the two axes of a plane (as defined by the **Normal Direction**).

- **Axial**

This type scales the surfaces about the **Origin** using a scaling **Factor** along a single **Axis**.

- **General**

This type scales the surfaces about the **Origin** using a separate scaling **Factor** for each of the defined axes. For 2D cases, you define one **Axis**, and the second axis is automatically defined as being perpendicular; for 3D cases, you define **Axis 1** and **Axis 2** (which should not be parallel), and the third axis is defined automatically as the cross product. Note that the surfaces will be scaled along each axis sequentially, so the order will be relevant for non-orthogonal axes.

For all types, the scaling **Factor** must be positive. A value of 1 results in no change, while a value greater or less than 1 results in stretching or shrinking, respectively. Note that you can enter a set value, or use the  button to assign an input parameter (for example, as part of a parametric study managed by Workbench); see [Select Input Parameter Dialog Box in the \*Fluent User's Guide\*](#) for details.

3. Click **Apply**.

## 2.7.6. Shape Modification

The procedure for computing the optimal design change and deforming the mesh if as follows:

1. Click the **Design Change** tab.
2. Select those boundaries that are free to be partially constrained or deformed in the **Zones To Be Modified** field. Boundaries that are not selected in this box that intersect the control volume will be constrained not to move. This is useful if there are walls that, for design reasons, must remain fixed and not move.
3. Select the constraining and/or deformation conditions ([Defining Conditions for the Deformation \(p. 49\)](#)) to be applied to the design change in the list of **Applied Conditions**. Note that a given surface cannot have multiple conditions of type **prescribed-profile, rotation, translation, or scaling**.
4. For each observable, specify its **Weight** in the optimization calculation. The weights are applied to the **Target/Reference Change** you specified in the **Objectives** tab. That is, if you have two observables that are each given a **Weight** equal to 1, Fluent will attempt to find an optimal design change that alters the observable values in proportion to their respective values for **Target/Reference Change**.

---

### Note

Weights are only required if there are two or more objectives that do not have explicit target changes specified.

---

5. Specify the deformation parameters.
  - a. Choose a **Freeform Scaling Scheme** and specify the **Freeform Scaling Factor**. This setting controls how the overall scaling factor for the design change magnitude is interpreted.

### Control-point spacing

The magnitude of the freeform deformation will be based on the control-point spacing. That is, a **Freeform Scale Factor** of 1.0 leads to maximum control point movement on the order of one sub-cell of the control volume. The sub-cell size is determined by a uniform sub-division of the morphed region based on the number of control points specified ([Defining Region Condi-](#)

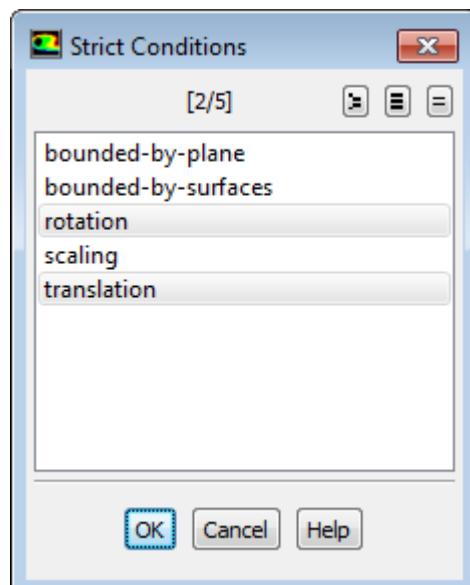
tions (p. 47)). A value of 1.0 is a reasonable default to preserve mesh quality during the morphing process.

### Objective reference change

The magnitude of the freeform deformation will be based on the **Target/Reference Change** value in the **Objectives** tab. That is, for a single observable optimization with **Freeform Scale Factor** of 1.0, the design change computed will yield an expected change in the observable equal to the **Target/Reference Change** for the observable.

- b. If you have disabled **Auto-Select Smoothness** in the **Numerics** tab (Design Tool Numerics (p. 56)) you can also specify a value for **Smoothness** of the deformed geometry. A lower value will allow locally sharper changes in the geometry, potentially at the expense of manufacturability.
  - c. If you used an input parameter as part of the setup of a deformation condition in the **Design Conditions** tab, the initial value will be displayed and is available for modification.
6. Click the **Check** button to verify that your constraining and/or deformation conditions are set up in such a way as to produce good results. A report will be printed in the console, summarizing the number of control points defined in the **Region Conditions** tab that are active (that is, not disqualified by other constraints) and listing conflicts between multiple constraints / deformations applied on a single node (including the **Design Conditions** and the continuity constraint applied along the boundary of the morphing region). The conflicts are categorized as being either of type **Fatal** (which indicates the solution is unlikely to converge or be desirable) or **Possible** (which may be allowable, depending on the details of the applied conditions).
  7. By default, the condition or conditions specified for a given zone (including the selected **Applied Conditions**, as well as the fixed condition applied to the unselected **Zones To Be Modified**) are not applied to every node, but instead to a subset of nodes distributed throughout the zone; this is to avoid overly constraining the zone. Consequently, some nodes will not strictly adhere to the condition, even if you have defined the constraint tolerance to be small or the residuals are small. To specify that all nodes of the zone strictly obey the condition, click the **Strict Conditions...** button and use the dialog box that opens (Figure 2.16: The Strict Conditions Dialog Box (p. 54)).

**Figure 2.16: The Strict Conditions Dialog Box**



Select the conditions for which you want strict enforcement and click **OK**. For unselected conditions, only the default subset of nodes will strictly obey the condition.

### Note

Strict enforcement is a direct modification of the surface mesh nodes to satisfy the design condition, and no modification is applied on the interior mesh. Consequently, when the boundary node modification is large, it may deteriorate the mesh or even cause negative volumes. Therefore you should use this condition carefully: the design condition should reach a good convergence and there should be enough control points.

When you click the **Modify Mesh** button (as described in a later step), the maximum offset for each condition (that is, the maximum distance a node has deviated from the prescribed condition) will be printed to the console.

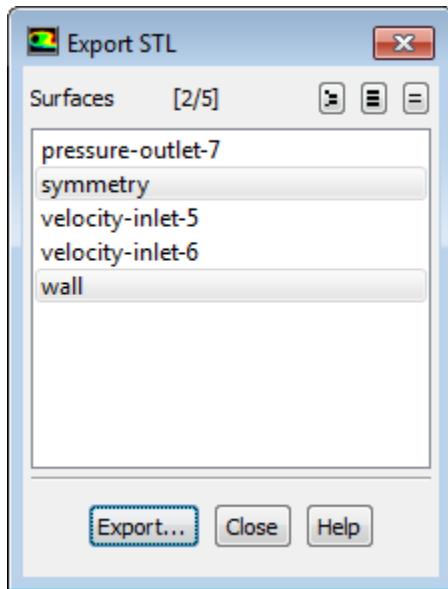
8. Click **Calculate Design Change** to compute the optimal design change. The **Expected change** for each observable will be reported in the **Design Tool** dialog box.
9. (optional) Click **Write Expected Change...** to write to file the adjustment in the observable that is expected due to the change defined by the current settings.
10. (optional) Click **Export Displacements...** to write to file the optimal surface displacement field (overwriting any pre-existing file of the same name). An example of the format of the text file that is written (for three-dimensions) is as follows:

```
Line 1: n
Lines 2 to n+1: x y z dx dy dz
```

11. The planned change to the geometry can be visualized by plotting the optimal displacement vector field or a contour plot of surface deformation.
12. Click **Modify Mesh** to deform the boundary and interior mesh with the optimal design change. If you determine that the change is unsatisfactory, you can click **Revert** to undo the mesh changes.

Upon successful modification of the geometry of the system, the modified shape can be viewed using the usual commands for viewing the mesh in ANSYS Fluent. After the mesh has been modified according to the control settings, the original flow calculation can be restarted and converged with the modified geometry. Upon successful convergence, the observable value should now have changed by an amount similar to that reported when the **Expected Change** button was clicked prior to deforming the geometry.

For 3D cases, you have the option of exporting the surfaces from your modified geometry as an .stl file, for use in other software packages. Click the **Export STL...** button near the bottom of the **Design Change** tab, select the **Surfaces** of interest in the dialog box that opens ([Figure 2.17: The Export STL Dialog Box \(p. 56\)](#)) and **Export...** them.

**Figure 2.17: The Export STL Dialog Box**

## 2.7.7. Design Tool Numerics

### **Smoothness**

As described in [Smoothing and Mesh Morphing \(p. 17\)](#), the mesh morphing is controlled by two superposed smoothing bases: the Bernstein polynomial basis which control the large scale smooth deformation and the B-spline basis which controls fine-scale deformation. The choice of smoothness determines the relative weighting of these bases and therefore the spatial scale of the design change. A large value of smoothness biases the design change to one in which optimal displacements vary in space in a manner dominated by the Bernstein polynomial basis. That is, they have a high degree of smoothness. A smaller value of smoothness leads to a recommended design change with smaller spatial scales.

The ability to vary the smoothness parameter is particularly important when prescribed displacements and design conditions are applied. Deformation at smaller spatial scales relieves the stiffness that would otherwise cause numerical challenges as well as lead to poor quality of the computed design and mesh.

By default, Fluent will automatically choose the smoothness. In the Numerics tab, you can disable **Auto-Select Smoothness** in which case a **Smoothness** input will be made available on the **Design Change** tab.

### **Calculation Numerics**

---

#### **Note**

If no **Design Conditions** are imposed, and if there are no walls that are treated as fixed which intersect the region being modified, the numerical settings described below are not used.

---

In many cases, no adjustment of the detailed numerics will be required for the design tool to compute an optimal solution. However, if the problem under investigation involves many Design Conditions and/or large displacements, the settings may need to be adjusted.

## Prescribed Motions

### Max. Iterations

The maximum number of inner iterations that will be performed in an effort to compute displacements from prescribed changes.

### Constraint Relaxation

The under-relaxation factor that is applied to the field to converge the prescribed displacements. This takes a value between 0 and 1.

### Preconditioning

A value of 0 yields the most aggressive advancement, while a larger value encourages more stable, albeit slower, advancement of the process for updating the prescribed displacements. A value of order 1 is typical in the event that preconditioning needs to be applied.

## Freeform Motions

### Max. Iterations

The maximum number of inner iterations that will be performed in an effort to compute freeform displacements that satisfy the applied **Design Conditions**.

### Constraint Relaxation

The under-relaxation factor that is applied to the field to converge the constrained freeform displacements. This takes a value between 0 and 1.

### Preconditioning

A value of 0 means that no preconditioning is applied, while a larger value encourages more stable, albeit slower, advancement of the process for updating the freeform displacements. A value of order 1 is typical in the event that preconditioning needs to be applied.

### Parameter Relaxation

For design conditions such as the Bounded-By-Plane condition there are additional parameters that are implicit in the calculation. The values of these parameters are computed as part of the solution process. The value of the parameter relaxation defines the rate at which these parameters are corrected as the calculation progresses. This takes a value between 0 and 1.

## Tolerances

### Constraints

The tolerance for convergence of the residuals associated with the design conditions themselves, or fixed zone conditions.

### Parameters

The tolerance for convergence of the parameters associated with the design conditions.

Using a value of 1 for **Constraint Relaxation** and a value of 0 for **Preconditioning** yields the most aggressive behavior. This can result in the fewest number of iterations to converge a constrained problem, at the expense of some added numerical effort and memory in preparing the preconditioning. For 2D and small 3D problems the added cost in time and memory is typically small. For larger problems involving multiple constraints a non-zero **Preconditioning** and **Constraint Relaxation** less than 1 may be preferable. The execution time to set up the preconditioning will be less, as will the memory usage. Calculation stability can also be improved. The number of iterations required to converge the problem will increase. However, this can prove to be an effective strategy that results in an overall shorter time needed to compute the design change.

## 2.8. Using the Adjoint Solver Module's Text User Interface

A text user interface (TUI) is provided for the adjoint solver. The root of the adjoint TUI is the `adjoint/` keyword at the top-most level of the command tree for ANSYS Fluent. The tree structure is as shown below:

**adjoint/observable**

Menu to create and configure observables of interest.

**create**

A new observable of the specified type and name is created and the definition is populated with default parameters.

**rename**

An existing observable is renamed to a new specified name.

**delete**

Removes a named observable.

**select**

A named observable is selected as the one for which an adjoint solution is to be computed.

**specify**

The parameters that define a named observable are configured.

**write**

Evaluates the current value of the selected observable and writes the result to a file.

**evaluate**

Evaluates the current value of the selected observable and prints the result to the console.

**adjoint/design-tool****design-conditions/**

Text menu to create, delete, rename, and set conditions on the geometry deformation.

**design-change/**

Text menu to perform optimal shape modification.

**calculate-design-change**

Compute the optimal design change.

**check**

Print a report in the console that summarizes the control points defined for the region, the defined constraining and/or deformation conditions, and any possible conflicts between multiple constraints / deformations applied on a single node.

**export-displacements**

Export the computed optimal displacements.

**export-stl**

Export specified surfaces from 3D cases as an .stl file.

**modify-mesh**

Apply the computed optimal displacement to the mesh.

**multi-objective-weights**

Set the weights for multiple free-form objectives.

**revert**

Revert (reject) the last mesh modification.

**select-conditions**

Select which conditions to apply to the mesh deformation.

**select-strict-conditions**

Select constraining and/or deformation conditions you want strictly enforced (that is, applied to all of the nodes of the associated zones).

**select-zones**

Select which zones are allowed to deform.

**settings**

Specify global deformation scale and settings.

**write-expected-changes**

Write out the expected changes in the observables for the computed optimal design change.

**numerics**

Adjust numerics settings for computing the optimal displacement.

**objectives/**

Text menu to configure observable objectives.

**include-current?**

Optionally include the most-recently computed observable sensitivity in the multi-objective design computation.

**manage/**

Text menu to import and manage previously exported sensitivity data.

**set**

Configure the objectives for loaded observables.

**region-conditions/**

Text menu to configure conditions on the deformation region

**boundaries**

Specify what degree of continuity to enforce at the boundaries of the deformation region.

**motion**

Specify conditions on the movement of the control points.

**points**

Specify the number of control points in the deformation region.

**symmetry**

Specify symmetry conditions for the deformation region.

**region/**

Text menu to define the deformation region.

**cartesian-limits**

Directly specify the bounds of the deformation region.

**export-sensitivities**

Export the sensitivity to mesh node movement for the currently-selected observable. Sensitivities are only exported for mesh nodes that lie within the specified deformation region.

**get-bounds**

Set the limits for the deformation region to a bounding box that encompasses a list of selected surfaces.

**larger-box**

Uniformly increase the size of the deformation region.

**smaller-box**

Uniformly decrease the size of the deformation region.

**adjoint/controls**

Menu to configure adjoint solver controls.

**settings**

Text menu to set parameters for the adjoint solver numerics.

**stabilization**

Text menu to set parameters for the adjoint stabilization schemes.

**adjoint/methods**

Menu to configure adjoint discretization settings.

**settings**

Text menu to set discretization methods for the adjoint solver.

**default-settings**

Sets discretization methods for the adjoint solver to the defaults.

**best-match-settings**

Sets discretization methods for the adjoint solver that best match with those for the flow solver.

**adjoint/monitors**

Menu to configure monitors for the adjoint solver.

**plot-residuals**

Plots the adjoint residuals in the designated graphics window.

**settings**

Text menu to configure the monitors and convergence criteria for the adjoint solver.

**adjoint/run**

Menu to initialize and compute the adjoint solution.

**initialize**

Initializes the adjoint solution field to zero everywhere.

**initialize-stabilization**

Initialize the stabilization data (only available if modal stabilization is used).

**iterate**

Advances the adjoint solver by a specified number of iterations, or until the convergence criteria are met.

**adjoint/reporting**

Menu to report sensitivity data from the adjoint solution.

**report**

Reports sensitivity data on a named flow boundary.

**write**

Reports sensitivity data on a named flow boundary and writes it to a named file.



---

## **Chapter 3:Tutorial:Using the Adjoint Solver – 2D Laminar Flow Past a Cylinder**

---

A tutorial is available that provides an example of how to generate sensitivity data for flow past a circular cylinder, how to postprocess the results, and how to use the data to perform a multi-objective design change that reduces drag and increases lift by morphing the mesh. The latest update of this tutorial is available on the ANSYS Customer Portal. To access tutorials and their input files on the ANSYS Customer Portal, go to <http://support.ansys.com/training>.



---

---

## **Part II: ANSYS Fluent Battery Module**

[Using This Manual \(p. lxvii\)](#)

- [1. Introduction \(p. 69\)](#)
- [2. Single-Potential Empirical Battery Model; \(p. 71\)](#)
- [3. Dual-Potential Empirical Battery Model \(p. 85\)](#)
- [4. Tutorial: Simulating a Single Battery Cell Using the MSMD Battery Model; \(p. 127\)](#)
- [5. Tutorial: Simulating a 1P3S Battery Pack Using the MSMD Battery Model; \(p. 129\)](#)

[Bibliography \(p. 131\)](#)

---

---

---

# Using This Manual

---

This preface is divided into the following sections:

**1. The Contents of This Manual**

## 1. The Contents of This Manual

The ANSYS Fluent Battery Module Manual provides information about using the battery models available in ANSYS Fluent. In this manual, you will find a theoretical discussion of the models used in ANSYS Fluent, and a description of using the models with your CFD simulations.

This part is divided into the following chapters:

- [Introduction \(p. 69\)](#)
- [Single-Potential Empirical Battery Model \(p. 71\)](#)
- [Dual-Potential MSMD Battery Model \(p. 85\)](#)
- [Tutorial: Simulating a Single Battery Cell Using the MSMD Battery Model \(p. 127\)](#)
- [Tutorial: Simulating a 1P3S Battery Pack Using the MSMD Battery Model \(p. 129\)](#)
- [Bibliography \(p. 191\)](#)



---

## Chapter 1: Introduction

---

The application of lithium ion batteries has been rapidly expanding from electric appliances and electronic devices to hybrid electric vehicles (HEVs) and electric vehicles (EVs), due to their high energy density. The main concerns when designing a Li-ion battery are its performance, life, and safety. The ANSYS Fluent battery models allow simulating a single battery cell or a battery pack using CFD technology to study their thermal and electrochemical behavior. The ANSYS Fluent battery models are provided as add-on modules with the standard ANSYS Fluent licensed software.

Note that the Fluent Tutorial Guide provides tutorials that illustrate how to use the ANSYS Fluent battery models. (To access tutorials and their input files on the ANSYS Customer Portal, go to <http://support.ansys.com/training>.)

This chapter contains the following information:

- 1.1. Overview
- 1.2. General Procedure
- 1.3. Installing the Battery Module

### 1.1. Overview

In a lithium-ion battery, the anode and cathode are made of active materials coated on the surface of metal foils. A polymer separator is placed between the foils of opposite polarity to prevent electrons from passing between them. To predict the evolution of the chemical, thermal, and electrical processes in a battery, ANSYS Fluent offers the following models:

- Single-Potential Empirical Battery Model (p. 71)
- Dual-Potential MSMD Battery Model (p. 85)

The Single-Potential Empirical Battery Model is useful if the geometries of the current collector, electrodes, and separator can be fully resolved. One potential equation is solved in the computational domain. This model is best suited for electrode-scale predictions in a single battery cell.

The model, however, has a limited ability to study the full range of electrochemical phenomena in battery systems, especially systems having complex geometry. When constructing a battery cell, the anode-separator-cathode sandwich layer is usually wound or stacked up into a 'jelly roll' or a prismatic shape. It would be very expensive to resolve all the layers explicitly, even for a single battery cell. Furthermore, many industrial applications use a battery pack consisting of a large number of cells connected in series or in parallel.

The ANSYS Fluent Dual Potential Multi-Scale Multi-Dimensional (MSMD) Battery model overcomes these limitations by using a homogeneous model based on a multi-scale multi-dimensional approach. In this approach, the whole battery is treated as an orthotropic continuum; thus, the mesh is no longer constrained by the micro-structure of the battery. Two potential equations are solved in the battery domain. To fit various analysis needs, the model includes three electrochemical submodels, namely, the Newman, Tiedemann, Gu, and Kim (NTGK) empirical model, the Equivalent Circuit model (ECM), and the Newman's Pseudo-2D (P2D) model having different level of complexity. The model offers you the flexibility to

study the physical and electrochemical phenomena that extend over many length scales in battery systems of various arrangements.

## 1.2. General Procedure

The following describes an overview of the procedure required in order to use the Battery Model in ANSYS Fluent.

1. Start ANSYS Fluent.
2. Read the mesh file.
3. Scale the grid, if necessary.
4. Load the module and use the **Battery Model** dialog box to define the battery model parameters.
5. Define material properties.
6. Set the operating conditions.
7. Set the boundary conditions.
8. Start the calculations.
9. Save the case and data files.
10. Process your results.

---

### Important

Note that the majority of this manual describes how to set up the ANSYS Fluent Battery Model using the graphical user interface. You can also perform various tasks using the text user interface.

---

## 1.3. Installing the Battery Module

The battery add-on modules are installed with the standard installation of ANSYS Fluent in a directories called `addons/battery` (for the single-potential empirical battery module) and `addons/msmdbatt` (for the dual-potential MSMD battery model) in your installation area. The battery modules consist of UDF libraries and pre-compiled scheme libraries that need to be loaded and activated before calculations can be performed. A number of UDFs are used to solve the battery equations. Once you loaded the battery add-on module, UDF and scheme libraries that are required by the battery model are *automatically* loaded. Further details are provided in chapters that describes the models.

---

## Chapter 2: Single-Potential Empirical Battery Model

---

This chapter discusses the theoretical background and the use of the Single-Potential Empirical Battery model with ANSYS Fluent. The information is organized in the following sections:

- 2.1. Single-Potential Empirical Battery Model Theory
- 2.2. Using the Single-Potential Empirical Battery Model

### 2.1. Single-Potential Empirical Battery Model Theory

This section describes the model theory that includes mathematical equations and physical interpretations of independent and dependent variables used in the model. The procedure for setting up and solving battery modeling is described in detail in [Using the Single-Potential Empirical Battery Model \(p. 73\)](#).

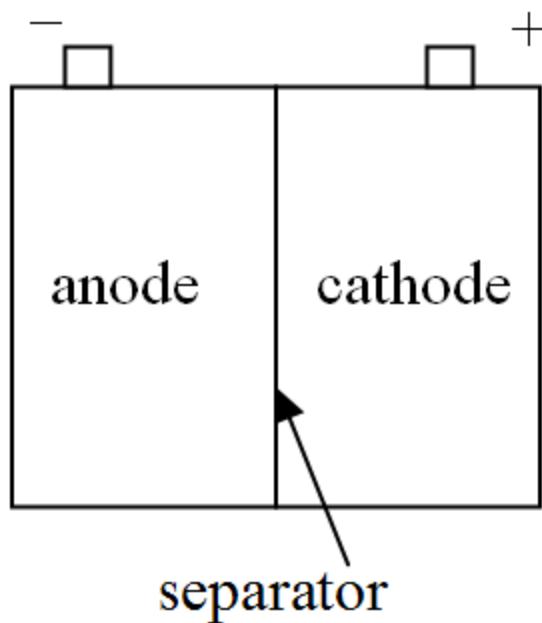
- 2.1.1. Introduction
- 2.1.2. Computation of the Electric Potential and Current Density
- 2.1.3. Thermal and Electrical Coupling

#### 2.1.1. Introduction

Given the important role of a battery in electric and/or hybrid electric vehicles there have been a number of models proposed in the literature to simulate the transient behavior of a rechargeable battery. These models vary in complexity from a zero-dimension resistor-capacitor circuit to a multi-dimension potential-current distribution, and, all the way to detailed electrochemistry modeling inside active separator layers. Computing resources (CPU time and memory) increase considerably with model complexity. Combining the need for model accuracy and the requirement for model usability, ANSYS Fluent has adopted a modeling approach that was based upon the 1D model initially proposed by Tiedemann and Newman [3] (p. 131), later used by Gu [1] (p. 131), and, more recently used by Kim et al [2] (p. 131) in their 2D study. ANSYS Fluent has extended the model formulation for use in 3D computations.

#### 2.1.2. Computation of the Electric Potential and Current Density

The computational domain includes only the anode and cathode electrodes and their current collectors. The separator layer is modeled as an infinitely thin interface between the two electrodes across which there is a potential jump due to loss. A cross-section of the anode-cathode assembly is schematically shown in the figure below.



The integral form of the electric potential equation reads,

$$\int_V \nabla \cdot (\sigma \nabla \phi) dV = \int_A j dA \quad (2.1)$$

In the above equation the source term  $j$ , also called the apparent current density, has non-zero value only at the anode-cathode interface (separator interface);  $A$  is the local surface area of the interface, and  $\sigma$  is the electric conductivity.

In their independent studies of small electrodes, Tiedemann and Newman [3] (p. 131), and Gu [1] (p. 131) observed that the apparent current density  $j$  ( $A/m^2$ ) varies linearly with cell voltage,

$$j = Y(\phi_c - \phi_a - U) \quad (2.2)$$

where the cell voltage is the difference between the cathode- and anode-side electric potentials at the separator interface ( $\phi_c - \phi_a$ ). From Equation 2.2 (p. 72) it is clear that on an experimentally measured polarization curve, namely the voltage-current (V-I) curve of a battery cell,  $U$  would be the intercept of  $V$  at  $I=0$ ; and,  $Y$  would be the inverse of the slope of the V-I curve. Moreover, Gu's experimental data [1] (p. 131) indicate that both  $U$  and  $Y$  vary with respect to the depth of discharge (DoD) defined relative to the theoretical battery capacity ( $Q_t$ ),

$$U = a_0 + a_1 DoD + a_2 DoD^2 + a_3 DoD^3 + a_4 DoD^4 + a_5 DoD^5 \quad (2.3)$$

$$Y = b_0 + b_1 DoD + b_2 DoD^2 + b_3 DoD^3 + b_4 DoD^4 + b_5 DoD^5 \quad (2.4)$$

where the coefficients  $a_i$  and  $b_i$  ( $i=0, \dots, 5$ ) are constants; and the local value of the depth of discharge is computed as follows,

$$DoD = \frac{A}{Q_t} \int_0^t j dt \quad (2.5)$$

## 2.1.3. Thermal and Electrical Coupling

While solution of [Equation 2.1 \(p. 72\)](#) to [Equation 2.5 \(p. 72\)](#) provides potential and the current density distributions in anode and cathode, the coupling between thermal and the electrical fields considers the following,

1. Temperature dependent electric conductivity  $\sigma=\sigma(T)$
2. Temperature dependent apparent current density ( $1/\Omega m^2$ ) by modifying [Equation 2.4 \(p. 72\)](#),

$$Y = \left( \sum_{n=0}^{n=0} b_n (DoD) \right)^n e^{C_1 (1/T_{ref} - 1/T)} \quad (2.6)$$

3. Temperature dependent equilibrium voltage ( $V$ ) by modifying [Equation 2.3 \(p. 72\)](#)

$$U = \left( \sum_{n=0}^{n=5} a_n (DoD)^n \right) - C_2 (T - T_{ref}) \quad (2.7)$$

4. Joule heating as a volumetric source term for thermal energy equation

$$S_{joule} = \frac{i^2}{\sigma} = \frac{\sigma^2 \nabla^2 \phi}{\sigma} = \sigma \nabla^2 \phi \quad (2.8)$$

5. Reaction heating as a volumetric source term for thermal energy equation,

$$S_{echem} = j \cdot [U - (\phi_c - \phi_a)] \frac{A}{Vol} \quad (2.9)$$

where  $Vol$  is the volume of the computing cell. Since the potential jump takes place only at the separator interface, the heat generated, [Equation 2.9 \(p. 73\)](#), is split equally between the two computing cells on either side of the interface.  $C_1$  and  $C_2$  are two additional model coefficients.

## 2.2. Using the Single-Potential Empirical Battery Model

The procedure for setting up and solving the Single-Potential Empirical battery model is described in detail in this section. Refer to [Single-Potential Empirical Battery Model Theory \(p. 71\)](#) for information about the theory.

- [2.2.1. Geometry Definition for the Single-Potential Empirical Battery Model](#)
- [2.2.2. Loading the Single-Potential Empirical Battery Module](#)
- [2.2.3. Getting Started With the Single-Potential Empirical Battery Model](#)
- [2.2.4. Solution Controls for the Single-Potential Empirical Battery Model](#)
- [2.2.5. Postprocessing the Single-Potential Empirical Battery Model](#)
- [2.2.6. User-Accessible Functions](#)
- [2.2.7. Using the Single-Potential Empirical Battery Model Text User Interface](#)

For information about inputs related to other models used in conjunction with the battery model, see the appropriate sections for those models in the ANSYS Fluent [User's Guide](#).

### 2.2.1. Geometry Definition for the Single-Potential Empirical Battery Model

Due to the fact that there are a number of different physical zones associated with the battery cell, the following regions must be present in the battery mesh:

- Anode

- Cathode
  - Separator ('zero' thickness wall/wall-shadow interface)
- 

**Note**

For electro-chemical types of simulation, 3D double-precision is recommended.

---

## 2.2.2. Loading the Single-Potential Empirical Battery Module

The single-potential empirical battery add-on module is installed with the standard installation of ANSYS Fluent in a directory called `addons/battery` in your installation area. The battery module consists of a UDF library and a pre-compiled scheme library, which need to be loaded and activated before calculations can be performed.

The battery module is loaded into ANSYS Fluent through the text user interface (TUI). The module can be loaded only when a valid ANSYS Fluent case file has been set or read. The text command to load the module is

```
define → models → addon-module
```

A list of ANSYS Fluent add-on modules is displayed:

```
Fluent Addon Modules:  
0. none  
1. MHD Model  
2. Fiber Model  
3. Fuel Cell and Electrolysis Model  
4. SOFC Model with Unresolved Electrolyte  
5. Population Balance Model  
6. Adjoint Solver  
7. Single-Potential Battery Model  
8. Dual-Potential MSMD Battery Model  
9. PEM Fuel Cell Model  
Enter Module Number: [0] 7
```

Select the single-potential battery model by entering the module number 7. During the loading process, a scheme library containing the graphical and text user interface, and a UDF library containing a set of user-defined functions (UDFs) for the battery module are loaded into ANSYS Fluent. This is reported to the console. The UDF library also becomes visible as a new entry in the **UDF Library Manager** dialog box. The basic setup of the battery model is performed automatically when the battery module is successfully loaded.

## 2.2.3. Getting Started With the Single-Potential Empirical Battery Model

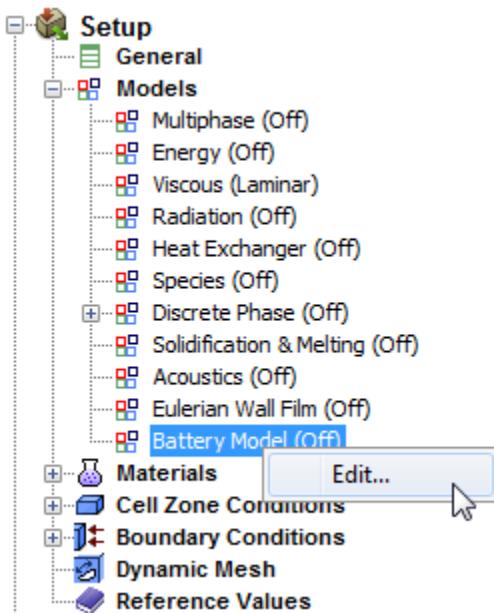
The battery model is implemented by user-defined functions (UDFs) and scheme routines in ANSYS Fluent. A number of UDFs are used to solve the battery equations. When you loaded the battery add-on module in the previous step ([Loading the Single-Potential Empirical Battery Module \(p. 74\)](#)), UDF and scheme libraries that are required by the battery model were *automatically* loaded. Before you can begin the process of defining your battery model, however, you will need to perform some additional setup tasks that involve allocating user-defined memory for the UDFs and hooking an adjust UDF to ANSYS Fluent. Follow the procedure below.

Once the module has been loaded, in order to set battery model parameters and assign properties to the relevant regions in your battery, you need to access the battery graphical user interface (the **Battery Model** dialog box).

To enable the battery model: In the tree under the **Models** branch, right-click **Battery Model** and click **Edit...** in the menu that opens.



**Figure 2.1: The Battery Model Option in the Tree**



This opens the **Battery Model** dialog box.



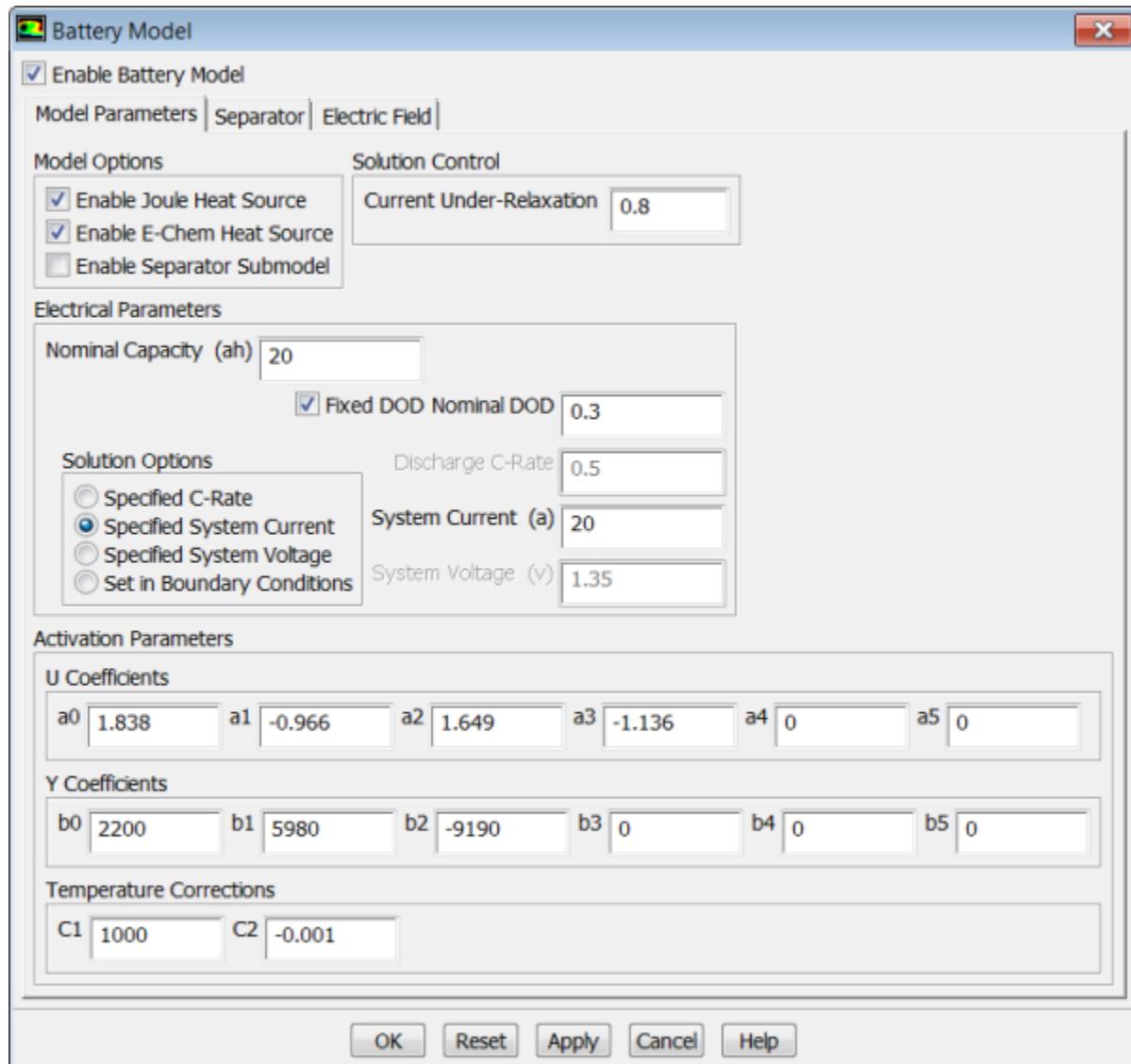
Once you open the **Battery Model** dialog box, you can select the **Enable Battery Model** check box to enable the model so that you can use it in your simulation. Enabling the model expands the dialog box to reveal additional model options and solution controls.

The **Model Parameters** tab of the **Battery Model** dialog box allows you to access general model settings when solving a battery problem. Likewise, the **Separator** tab allows you to set options for the battery separator. Finally, the **Electric Field** tab allows you to set parameters for the electric field. For additional information, see the following sections:

- [2.2.3.1. Specifying Single-Potential Empirical Battery Model Parameters](#)
- [2.2.3.2. Specifying Separator Parameters](#)
- [2.2.3.3. Specifying Electric Field Parameters](#)

### **2.2.3.1. Specifying Single-Potential Empirical Battery Model Parameters**

The **Model Parameters** tab of the **Battery Model** dialog box allows you to turn on or off various options when solving a battery problem.

**Figure 2.2: The Battery Model Dialog Box (Model Parameters Tab)**

In the **Model Parameters** tab, you can set various model options, solution controls, electrical parameters, as well as activation parameters.

For **Model Options**, you can:

- **Enable Joule Heat Source** in order to include the Joule Heating source in the thermal energy equation (Equation 2.8 (p. 73)). (enabled by default)
- **Enable E-Chem Heat Source** in order to include the heat source due to electrochemistry in the thermal energy equation (Equation 2.9 (p. 73)). (enabled by default)
- **Enable Separator Submodel** in order to specify your own values for the **Separator Thickness** and **Resistivity**, instead of the **Y Coefficients**, for modeling the separator. Note that when this option is selected, the

inputs for the **Y Coefficients** are grayed out, and the **Separator Property** fields are active in the **Separator** tab.

---

### Note

The separator submodel is provided as an alternative method to calculate the separator resistance. When the option is enabled, the separator resistance is computed as  $R = (\text{separator resistivity}) * (\text{separator thickness})$ , where *separator resistivity* and *separator thickness* are supplied by you. When the option is disabled (the default), the separator resistance is computed by  $R=1/Y$ .

---

For **Solution Controls**, you can set the **Current Under-Relaxation Factor**.

For **Electrical Parameters**, you can set the **Nominal Capacity** (the capacity of the battery cell). If you select **Fixed DoD** (for steady state simulation only), then you can specify a **Nominal DoD** value (depth of discharge). For the **Solution Options**, if you select:

- **Specified C-Rate**, you can set a value for the **Discharge C-Rate** (the hourly rate at which a battery is discharged). In this case, the total current at the cathode tabs are fixed as the product of C-Rate and Nominal Capacity, while the electrical potential is anchored at zero on the anode tabs.
  - **Specified System Current**, you can set a value for the total current (applied to the anode tabs). In this case, the electrical potential is set to zero at the anode tabs.
  - **Specified System Voltage**, you can set a value for the **System Voltage** (applied to the cathode tab; the anode tab has a voltage of 0 V).
  - **Set in Boundary Conditions**, you can set the UDS boundary conditions directly, for example, the voltage value or the current value (specified flux), using the **Boundary Conditions** task page in ANSYS Fluent for the specific face zone.
- 

### Note

When the steady state solver is used, the **Fixed DoD** option has to be selected. Alternatively, the transient solver can be used to analyze variable DoD problems.

---

For **Activation Parameters**, you can specify the **U Coefficients** for [Equation 2.3 \(p. 72\)](#) and the **Y Coefficients** for [Equation 2.4 \(p. 72\)](#) (if the **Enable Separator Submodel** option is disabled).

---

### Note

The coefficient values for the **Activation Parameters** are based on battery cell polarization test curves. Obtaining coefficient values (other than the default values) can be dependant on your battery configuration and material properties. For more information about coefficient values, refer to the work performed by Gu [1] ([p. 131](#)). You will likely need to make adjustments (for example, if you are modeling lithium ion batteries) when using your own experimental data.

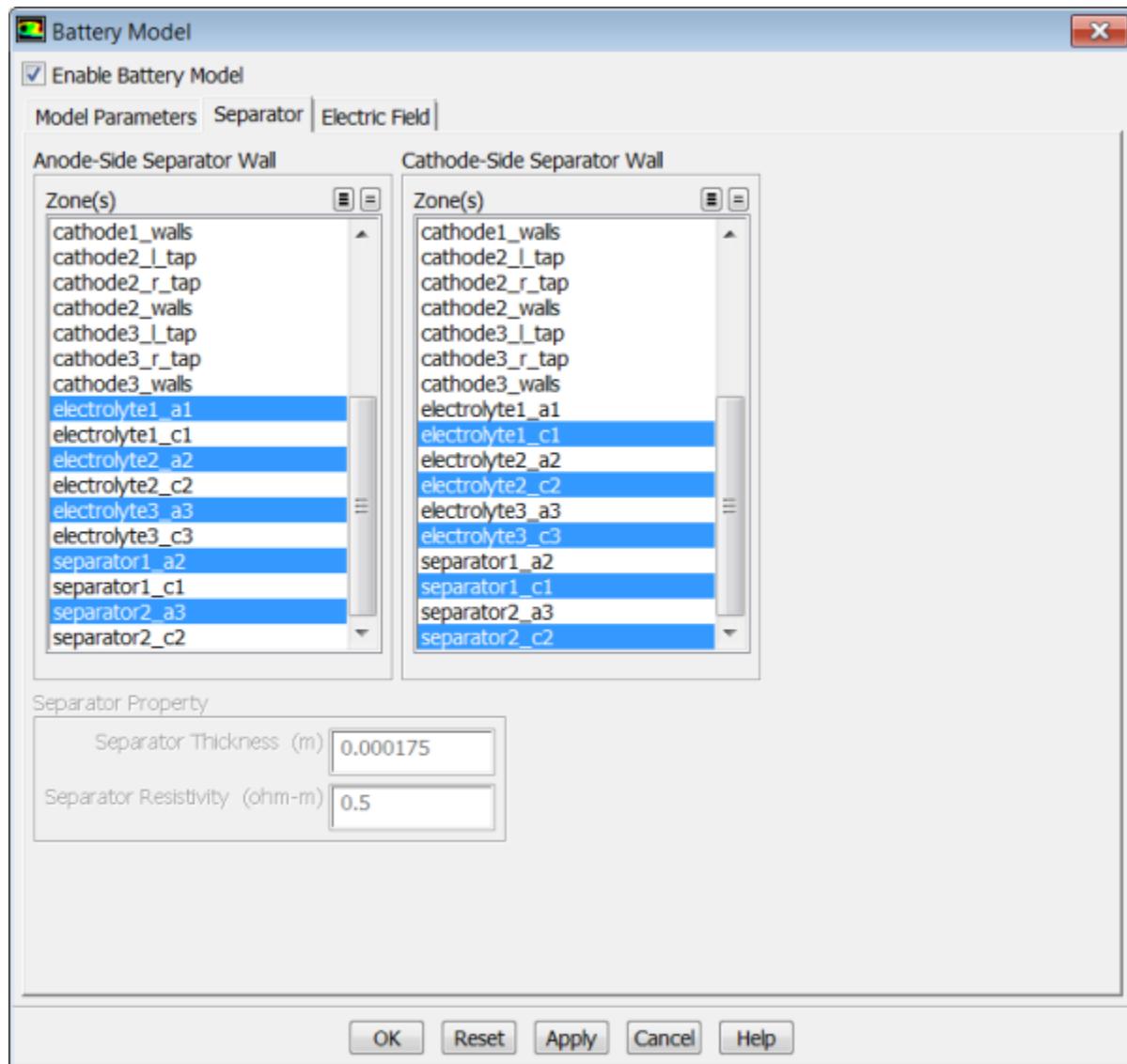
---

You can also specify the **Temperature Corrections**, if needed, though the default values are suitable in most cases. The temperature corrections provide additional accuracy to account for local temperature effects, and correspond to the temperature terms in [Equation 2.6 \(p. 73\)](#) and [Equation 2.7 \(p. 73\)](#).

### 2.2.3.2. Specifying Separator Parameters

The **Separator** tab of the **Battery Model** dialog box allows you to select interfaces as the **Anode Separator**, the **Cathode Separator**, as well as the **Separator Properties**, if appropriate.

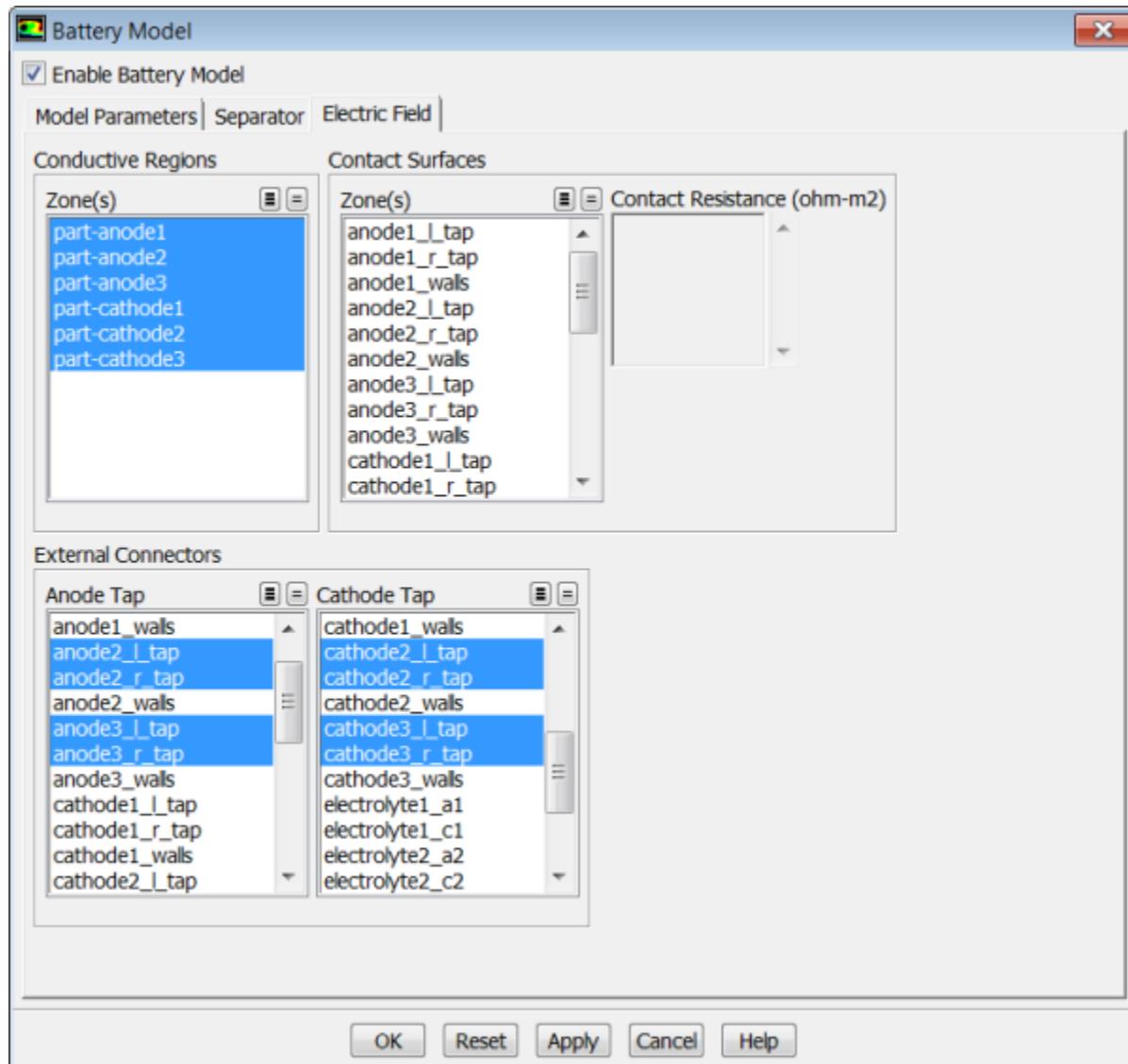
**Figure 2.3: The Battery Model Dialog Box (Separator Tab)**



In the **Separator** tab, specify the zones for the **Anode Separator** and the **Cathode Separator**. If the **Enable Separator Submodel** option is selected in the **Model Parameters** tab, you can specify the **Separator Properties** such as the **Separator Thickness** and the **Separator Resistivity**.

### 2.2.3.3. Specifying Electric Field Parameters

The **Electric Field** tab of the **Battery Model** dialog box allows you to set the properties of the **Conductive Regions**, **Contact Surfaces**, and the **External Connectors**.

**Figure 2.4: The Battery Model Dialog Box (Electric Field Tab)**

In the **Electric Field** tab, specify the zones for the **Conductive Regions** and the **Contact Surfaces** (as well as the **Contact Resistance**) for any selected contact surface. In addition, you can specify the anode and cathode tap surfaces for the External Connectors of the battery.

## 2.2.4. Solution Controls for the Single-Potential Empirical Battery Model

When you use the Battery model, the electric potential equation is solved in addition to other fluid dynamic equations, depending on the type of simulation. The electric potential equation is listed as one of the solved equations by ANSYS Fluent in the **Equation** dialog box, where it can be enabled/disabled in the solution process.

**Solution** → **Solution Controls** → **Equations...**

Also, keep in mind the **Advanced Solution Controls** dialog box, where you can set the multigrid cycle type for the electric potential equation, if required.

 **Setup** →  **Solution Controls** → **Advanced...**

---

### Note

When choosing a solution method for your simulation, the Least Squares Cell Based gradient spatial discretization method is recommended because of its greater accuracy. The Green-Gauss Cell Based gradient spatial discretization method is adequate if the mesh is evenly spaced in the system current direction, and if there are not large differences in the electrical conductivities in the materials used in the simulation.

---

### Note

In transient simulations, you should start the calculation with a smaller time step (1 ~ 2 seconds) initially. The time step can be increased to a large value (for example, 30 seconds), however, you will likely need to search for a suitable value to make sure reasonable convergence is achieved within each time step.

---

## 2.2.5. Postprocessing the Single-Potential Empirical Battery Model

You can perform post-processing using standard ANSYS Fluent quantities and by using user-defined scalars and user-defined memory allocations. When using the Battery model, the following additional variables will be available for postprocessing:

- Under **User-Defined Scalars...**
  - **Electric Potential**
  - **Diff Coef of Electric Potential** (the electrical conductivity of the conductive field)
- Under **User-Defined Memory...**
  - **Interface Current Density** (the separator current density, that is J (A/m<sup>2</sup>))
  - **X Current Density**
  - **Y Current Density**
  - **Z Current Density**
  - **Magnitude of Current Density**
  - **Volumetric Ohmic Source** (the energy source due to the electric Joule heating)
  - **Electrochemistry Source**
  - **Activation Over-Potential** (the net electrode potential change across the anode and cathode of the system when there is a current flowing through the system, that is,  $\phi_c - \phi_a - U$  (Volts))
  - **Depth of Discharge**
  - **U Function**
  - **Y Function**

- **Separator Voltage Jump** (the net potential difference across the separator)
- **Effective Electric Resistance** (the effective resistance of the separator used in the potential field calculation)
- **Other**

By default, the ANSYS Fluent Battery Model defines several user-defined scalars and user-defined memory allocations, described in [Table 2.1: User-Defined Scalar Allocations \(p. 81\)](#) and [Table 2.2: User-Defined Memory Allocations \(p. 81\)](#).

**Table 2.1: User-Defined Scalar Allocations**

UDS 0	Electric Potential (Volts)
UDS 1	Diff Coef of Electric Potential

**Table 2.2: User-Defined Memory Allocations**

UDM 0	Interface Current Density
UDM 1	X Current Density
UDM 2	Y Current Density
UDM 3	Z Current Density
UDM 4	Magnitude of Current Density
UDM 5	Volumetric Ohmic Source
UDM 6	Electrochemistry Source
UDM 7	Activation Over-Potential
UDM 8	Depth of Discharge
UDM 9	U Function
UDM 10	Y Function
UDM 11	Separator Voltage Jump
UDM 12	Effective Electric Resistance
UDM 13	Other

### Note

- For field variables that are stored in UDM, use the corresponding variables for post processing. Post processing the UDM itself is not recommended.
- For reviewing simulation results in CFD Post for cases involving UDM or UDS, export the solution data as CFD-Post-compatible file (.cda.t) in Fluent and then load this file into CFD-Post.

## 2.2.6. User-Accessible Functions

You can incorporate your own formulations and data for the properties of the battery using the `batt_user.c` source code file.

The following listing represents a description of the contents of the `batt_user.c` source code file:

- real CONDUCTIVITY\_CELL(cell\_t c, Thread \*t): Returns values for the electrical conductivity for cells inside conductive zones, overwriting the values set in the **Electric Field** tab of the **Battery Model** dialog box.
- real CONTACT\_RESIST\_FACE(cell\_t f, Thread \*t): Returns values for the electrical contact resistance, overwriting the values set in the **Electric Field** tab of the **Battery Model** dialog box.
- real Compute\_U (face\_t f, Thread \*t, real DOD, real dUDJ, real a[], real \*dUDJ): Returns values for both U and the derivative of U with respect to J (dUDJ). Note that DOD represents the depth of discharge and a[] is the set of U coefficients.
- real Compute\_Y (face\_t f, Thread \*t, real DOD, real b[]): Returns values for Y. Note that DOD represents the depth of discharge and b[] is the set of Y coefficients.

For more information, see the following sections:

[2.2.6.1. Compiling the Customized Battery Source Code](#)

## **2.2.6.1. Compiling the Customized Battery Source Code**

This section includes instructions on how to compile a customized Battery user-defined module. Note that you can also refer to the file INSTRUCTIONS-CLIENT that comes with your distribution (see addons/battery).

---

### **Important**

It is assumed that you have a basic familiarity with compiling user-defined functions (UDFs). For an introduction on how to compile UDFs, refer to the [Fluent Customization Manual](#).

---

You will first want to use a local copy of the battery directory in the addons directory before you recompile the Battery module.

### **2.2.6.1.1. Compiling the Customized Source Code Under Linux**

1. Make a local copy of the battery directory. Do not create a symbolic link.
- 

### **Important**

The custom version of the library must be named according to the convention used by ANSYS Fluent: for example, battery.

---

2. Change directories to the battery/src directory.
3. Make changes to the batt\_user.c file.
4. Edit the makefile located in the src/ directory and make sure that the FLUENT\_INC variable correctly refers to the current ANSYS Fluent installation directory. Be careful not to leave any trailing spaces when you make your changes.
5. Define the FLUENT\_ADDONS environment variable to correspond to your customized version of the Battery module.
6. Change directories to the battery/ directory.

7. Issue the following make command:

```
make FLUENT_INC=[ansys_inc/v170/fluent] FLUENT_ARCH=[arch]
-f Makefile-client
```

where `your_arch` is `lnx86` on `LINUX`, or `ultra` on the `Sun` operating system, etc.

The following example demonstrates the steps required to set up and run a customized version of the `Battery` module that is located in a folder call `home/sample`:

- Make a directory (for example, `mkdir -p /home/sample`).
- Copy the default addon library to this location.

```
cp -RH [ansys_inc/v170/fluent]/fluent17.0.0/addons/battery
/home/sample/battery
```

- Using a text editor, make the appropriate changes to the `batt_user.c` file located in `/home/sample/battery/src/batt_user.c`
- Edit the `makefile` located in the `src/` directory and make sure that the `FLUENT_INC` variable correctly refers to the current ANSYS Fluent installation directory. Be careful not to leave any trailing spaces when you make your changes.
- Build the library.

```
cd /home/sample/battery
make FLUENT_INC=[ansys_inc/v170/fluent] FLUENT_ARCH=[arch]
-f Makefile-client
```

- Set the `FLUENT_ADDONS` environment variable (using CSH, other shells will differ).

```
setenv FLUENT_ADDONS /home/sample
```

- Start ANSYS Fluent and load the customized module using the text interface command.

### **2.2.6.1.2. Compiling the Customized Source Code Under Windows**

1. Open **Visual Studio.NET** at the DOS prompt.
2. Make sure that the `FLUENT_INC` environment variable is correctly set to the current ANSYS Fluent installation directory (for example, `ANSYS Inc\v170\fluent`).
3. Make a local copy of the `battery` folder. Do not create a shortcut.
4. Enter the `battery\src` folder.
5. Make changes to the `batt_user.c` file.
6. Define the `FLUENT_ADDONS` environment variable to correspond to your customized version of the `Battery` module.
7. Return to the `battery` folder.
8. Issue the following command in the command window:

```
nmake /f makefile_master-client.nt
```

## 2.2.7. Using the Single-Potential Empirical Battery Model Text User Interface

All of the features for the Battery Model that are available through the graphical user interface are also available through text user interface (TUI) commands. The TUI allows text commands to be typed directly in the ANSYS Fluent console window where additional information can be extracted and processed for more advanced analysis.

Once the battery module is loaded, you can access the text user interface through the Console Window under `battery-model`. A listing of the various text commands is as follows:

**battery-model/**

Battery model menu

**activation-parameters/**

Activation parameter setup.

**t-coefficients**

Specify the temperature coefficients in [Equation 2.6 \(p. 73\)](#) and [Equation 2.7 \(p. 73\)](#).

**u-coefficients**

Specify the U coefficients for [Equation 2.3 \(p. 72\)](#).

**y-coefficients**

Specify the Y coefficients for [Equation 2.4 \(p. 72\)](#).

**anode-interface**

Anode interface options.

**cathode-interface**

Cathode interface options.

**electric-field-model/**

Electric field setup.

**conductive-regions**

List zone names and IDs.

**contact-resistance-regions**

List zone names and IDs.

**current-tap**

List zone names and IDs.

**voltage-tap**

List zone names and IDs.

**electrochemistry**

Electrochemistry parameters.

**enable-battery-model?**

Enable/disable battery model.

**model-parameters**

Model parameters.

---

## Chapter 3: Dual-Potential MSMD Battery Model

---

This chapter discusses how you can model the battery using Dual Potential Multi-Scale Multi-Domain (MSMD) approach. It includes a summary of the theory and implementation of three electrochemical submodels that are available in ANSYS Fluent. The information is organized in the following sections:

- 3.1. Dual-Potential MSMD Battery Model Theory
- 3.2. Using the Dual-Potential MSMD Battery Model

### 3.1. Dual-Potential MSMD Battery Model Theory

This section provides an overview and theoretical background of the Multi-Scale Multi-Domain (MSMD) method. The procedure for setting up and solving battery modeling is described in detail in [Using the Dual-Potential MSMD Battery Model \(p. 97\)](#).

- 3.1.1. MSMD approach
- 3.1.2. NTGK Model
- 3.1.3. ECM Model
- 3.1.4. Newman's P2D Model
- 3.1.5. Coupling Between CFD and Submodels
- 3.1.6. Battery Pack Simulation
- 3.1.7. Reduced Order Solution Method (ROM)
- 3.1.8. External and Internal Electric Short-Circuit Treatment
- 3.1.9. Thermal Abuse Model

#### 3.1.1. MSMD approach

The difficulty with modeling a lithium-ion (Li-ion) battery is due to its multi-domain, multi-physics nature. Vastly different length scales associated with different physics complicates the problem. When performing a thermal analysis, the goal is to determine the temperature distribution at the battery length scale. The physics governing the Li-ion transport occurs in the anode-separator-cathode sandwich layers (the electrode pair length scale). Li-ion transport in an active material occurs at the atomic length scale. The Multi-Scale Multi-Domain (MSMD) approach deals with different physics in different solution domains [\[\[8\] \(p. 131\)\]](#).

Battery thermal and electrical fields are solved in the CFD domain at the battery cell's scale using the following differential equations:

$$\frac{\partial \rho C_p T}{\partial t} - \nabla \cdot (k \nabla T) = \sigma_+ |\nabla \phi_+|^2 + \sigma_- |\nabla \phi_-|^2 + \dot{q}_{ECh} + \dot{q}_{short} + \dot{q}_{abuse} \quad (3.1)$$

$$\begin{aligned} \nabla \cdot (\sigma_+ \nabla \phi_+) &= - (j_{ECh} - j_{short}) \\ \nabla \cdot (\sigma_- \nabla \phi_-) &= j_{ECh} - j_{short} \end{aligned} \quad (3.2)$$

where  $\sigma_+$  and  $\sigma_-$  are the effective electric conductivities for the positive and negative electrodes,  $\phi_+$  and  $\phi_-$  are phase potentials for the positive and negative electrodes,  $j_{ECh}$  and  $\dot{q}_{ECh}$  are the volumetric current transfer rate and the electrochemical reaction heat due to electrochemical reactions, respectively,  $j_{short}$  and  $\dot{q}_{short}$  are the current transfer rate and heat generation rate due to battery internal short-circuit,

respectively, and  $\dot{q}_{abuse}$  is the heat generation due to the thermal runaway reactions under the thermal abuse condition.

For normal operation,  $\dot{q}_{abuse}$  is set to zero. For more information, refer to [Thermal Abuse Model \(p. 95\)](#). The source terms  $j_{ECh}$  and  $\dot{q}_{ECh}$  are computed using an electrochemical submodel. If there is no internal short-circuit,  $j_{short}$  and  $\dot{q}_{short}$  are equal to zero. For more information, refer to [External and Internal Electric Short-Circuit Treatment \(p. 94\)](#).

A wide range of electrochemical models, from simple empirically-based to fundamental physics-based, is available in the open literature. In ANSYS Fluent, the following electrochemical submodels are implemented:

- Newman, Tiedemann, Gu and Kim (NTGK) model
- Equivalent Circuit Model (ECM) model
- Newman Pseudo-2D (P2D) model

In addition, you can define your own electrochemical model via user-defined functions and hook it to the Fluent MSMD battery module.

To use the MSMD approach in an arbitrary finite volume of the cell composite, the following conditions must be met:

- Battery micro layers must have the same orientation
- Two potential fields,  $\varphi_+$  and  $\varphi_-$ , must be uniquely determined

These conditions are usually met for aligned stack cells or wound cells with extended foil-type continuous current tabs.

### 3.1.2. NTGK Model

The Newman, Tiedemann, Gu, and Kim (NTGK) model is a simple semi-empirical electrochemical model. It was proposed by Kwon [\[\[11\] \(p. 131\)\]](#) and has been used by others [\[\[10\] \(p. 131\), \[9\] \(p. 131\)\]](#). In the model formulation, the volumetric current transfer rate in [Equation 3.2 \(p. 85\)](#) is related to the potential field by the following algebraic equation:

$$j_{ECh} = aY[U - (\varphi_+ - \varphi_-)] \quad (3.3)$$

where  $a$  is the specific area of the electrode sandwich sheet in the battery,  $Y$  and  $U$  are the model parameters which are functions of the battery depth of discharge (DoD):

$$DoD = \frac{Vol}{3600Q_{Ah}} \left( \int_0^t j dt \right) \quad (3.4)$$

where  $Vol$  denotes the battery volume, and  $Q_{Ah}$  is the battery total electric capacity in Ampere hours.

For a given battery, the voltage-current response curve can be obtained through experimentation.  $Y$  and  $U$  can then be determined by curve fitting the data. ANSYS Fluent has adopted the following formulation for the  $Y$  and  $U$  functions proposed in [\[\[11\] \(p. 131\)\]](#):

$$Y = \left( \sum_{n=0}^{\infty} a_n (DoD)^n \right) \exp \left[ -C_1 \left( \frac{1}{T} - \frac{1}{T_{ref}} \right) \right] \quad (3.5)$$

$$U = \left( \sum_{n=0}^{\infty} b_n (DoD)^n \right) - C_2 (T - T_{ref})$$

where  $C_1$  and  $C_2$  are the battery-specific NTGK model constants.

The electrochemical reaction heat  $\dot{q}_{ECH}$  in Equation 3.1 (p. 85) is calculated as

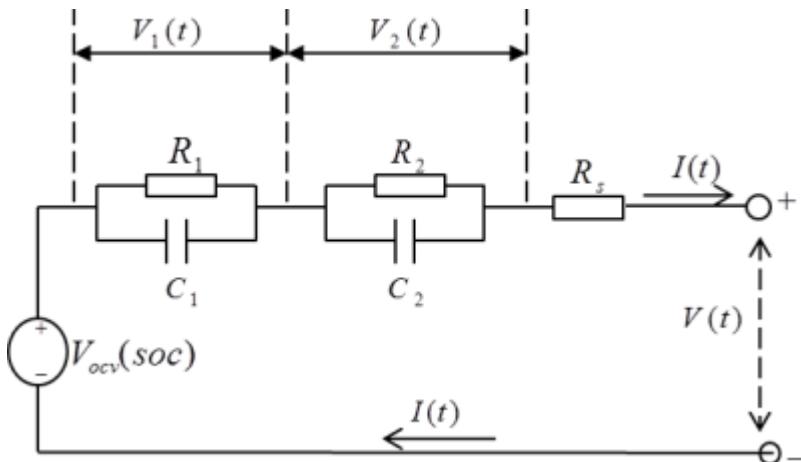
$$\dot{q}_{ECH} = j_{ECH} \left[ U - (\varphi_+ - \varphi_-) - T \frac{dU}{dT} \right] \quad (3.6)$$

where the first term is heat due to overpotential and the second term is heat due to entropic heating.

### 3.1.3. ECM Model

In the Equivalent Circuit Model (ECM), battery electric behavior is mimicked by an electrical circuit. ANSYS Fluent has adopted the six parameter ECM model following the work of Chen [[5] (p. 131)]. In this model, the circuit consists of three resistors and two capacitors (Figure 3.1: Electric Circuits Used in the ECM Model (p. 87)).

**Figure 3.1: Electric Circuits Used in the ECM Model**



The voltage-current relation can be obtained by solving the electric circuit equations:

$$V(t) = V_{OCV}(soc) - V_1 - V_2 - R_s(soc)I(t) \quad (3.7)$$

$$\frac{dV_1}{dt} = -\frac{1}{R_1(soc)C_1(soc)} V_1 - \frac{1}{C_1(soc)} I(t)$$

$$\frac{dV_2}{dt} = -\frac{1}{R_2(soc)C_2(soc)} V_2 - \frac{1}{C_2(soc)} I(t)$$

$$\frac{d(soc)}{dt} = I(t) / 3600Q_{Ah}$$

For a given battery, the open circuit voltage, resistors' resistances, and capacitors' capacitances are functions of the battery state of charge (SOC). These functions could be expressed in two different ways in ANSYS Fluent:

- The fifth order Polynomial form:

$$R_s = a_0 + a_1(soc) + a_2(soc)^2 + a_3(soc)^3 + a_4(soc)^4 + a_5(soc)^5 \quad (3.8)$$

$$R_1 = b_0 + b_1(soc) + b_2(soc)^2 + b_3(soc)^3 + b_4(soc)^4 + b_5(soc)^5$$

$$C_1 = c_0 + c_1(soc) + c_2(soc)^2 + c_3(soc)^3 + c_4(soc)^4 + c_5(soc)^5$$

$$R_2 = d_0 + d_1(soc) + d_2(soc)^2 + d_3(soc)^3 + d_4(soc)^4 + d_5(soc)^5$$

$$C_2 = e_0 + e_1(soc) + e_2(soc)^2 + e_3(soc)^3 + e_4(soc)^4 + e_5(soc)^5$$

$$V_{ocv} = f_0 + f_1(soc) + f_2(soc)^2 + f_3(soc)^3 + f_4(soc)^4 + f_5(soc)^5$$

- The function form proposed by Chen [[5] (p. 131)]:

$$\begin{aligned} R_s &= a_0 + a_1 \exp[-a_2(soc)] \\ R_1 &= b_0 + b_1 \exp[-b_2(soc)] \\ C_1 &= c_0 + c_1 \exp[-c_2(soc)] \\ R_2 &= d_0 + d_1 \exp[-d_2(soc)] \\ C_2 &= e_0 + e_1 \exp[-e_2(soc)] \\ V_{ocv} &= f_0 + f_1(soc) + f_2(soc)^2 + f_3(soc)^3 + f_4 \exp[-f_5(soc)] \end{aligned} \quad (3.9)$$

The source terms for [Equation 3.1 \(p. 85\)](#) and [Equation 3.2 \(p. 85\)](#) are computed as:

$$j_{ECh} = I / Vol \quad (3.10)$$

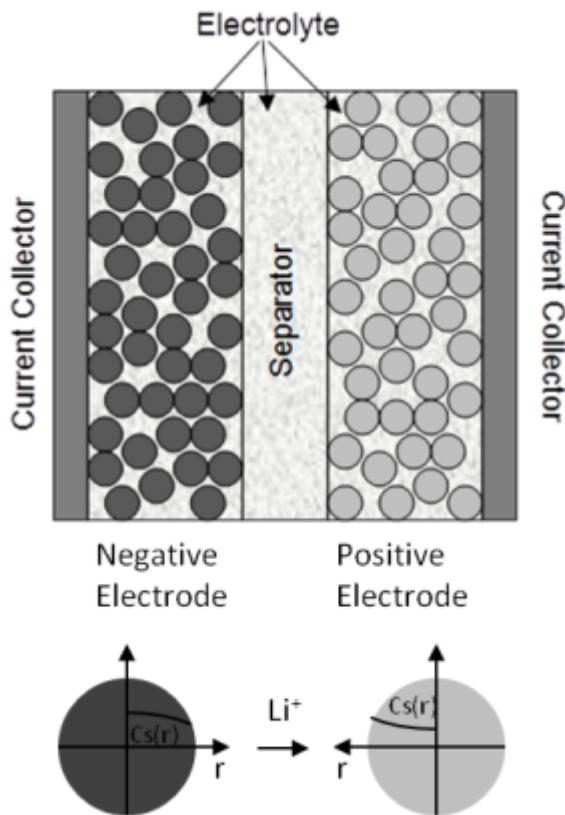
$$\dot{q}_{ECh} = \frac{I}{Vol} \left[ V_{ocv} - (\varphi_+ - \varphi_-) - T \frac{dU}{dT} \right] \quad (3.11)$$

where  $Vol$  denotes the battery volume,  $I$  is the current, and  $V_{ocv}$  is the open circuit voltage.

### 3.1.4. Newman's P2D Model

Newman's group developed a physics based model using porous electrode and concentrated solution theories [[6] (p. 131)]. The model can accurately capture Li-ion migration in the battery. It has been widely used in the literature [[12] (p. 131), [4] (p. 131)].

[Figure 3.2: Electrode and Particle Domains in the Newman's Model \(p. 89\)](#) schematically shows the electrode plate pair in the Li-ion battery. The composite electrode consists of active materials and electrolyte solution. The electrolyte phase is continuous across the negative electrode, separator and positive electrode while solid phase only exists in the negative and positive electrode. The solid active material is modeled as a matrix of mono-sized spherical particles.

**Figure 3.2: Electrode and Particle Domains in the Newman's Model**

During the discharge process, Li diffuses to the surface of negative electrode particles and undergoes an electrochemical reaction. This reaction releases an electron and transfers Li to the electrolyte phase. The Li-ions diffuse and conduct through the electrolyte solution from the negative electrode to the positive electrode, where a similar reaction transfers Li to the positive solid phase. Li is stored inside the positive electrode particles until the cell is later recharged.

The Li-ion transport phenomena in the porous electrode and electrolyte can be described by the charge and mass conservation laws. Charge conservation governs phase potentials,  $\varphi_+$  and  $\varphi_-$ , while mass conservation governs the phase concentrations,  $c_e$  and  $c_s$ , where the subscripts  $e$  and  $s$  are used to denote the electrolyte and solid (electrode) phases, respectively.

The conservation equations relevant to electrolyte and solid phases are described as:

- Lithium conservation in the solid phase:

$$\frac{\partial c_s}{\partial t} = \frac{D_s}{r^2} \frac{\partial}{\partial r} \left( r^2 \frac{\partial c_s}{\partial r} \right) \quad (3.12)$$

- Lithium conservation in the electrolyte phase:

$$\frac{\partial (\varepsilon_e c_e)}{\partial t} = \frac{\partial}{\partial x} \left( D_e^{eff} \frac{\partial c_e}{\partial x} \right) + \frac{1-t_+^0}{F} j^{Li} \quad (3.13)$$

- Charge conservation in solid phase:

$$\frac{\partial}{\partial x} \left( \sigma^{eff} \frac{\partial \varphi_s}{\partial x} \right) - j^{Li} = 0 \quad (3.14)$$

- Charge conservation in the electrolyte phase:

$$\frac{\partial}{\partial x} \left( k^{eff} \frac{\partial \varphi_e}{\partial x} \right) + \frac{\partial}{\partial x} \left( k_D^{eff} \frac{\partial \ln c_e}{\partial x} \right) + j^{Li} = 0 \quad (3.15)$$

The Lithium conservation diffusion equation needs to be solved at every discretized spatial location in the electrode zone. The Lithium conservation equation is solved in the r-dimension of the spherical particles—the pseudo second dimension. That is why the model is often referred to as the Newman's P2D model in the literature.

The Butler-Volmer equation is used to couple the charge and species governing equations by describing  $j^{Li}$  as a function of overpotential,  $\eta$ :

$$j^{Li} = a_s i_0 \left\{ \exp\left(\frac{\alpha_a F}{RT} \eta\right) - \exp\left(-\frac{\alpha_c F}{RT} \eta\right) \right\} \quad (3.16)$$

where  $\eta$  is defined by

$$\eta = \varphi_s - \varphi_e - U \quad (3.17)$$

and  $i_0$  is the exchange current density defined by

$$i_0 = k_m (c_e)^{\alpha_a} (C_{s,max} - c_{s,e})^{\alpha_a} (c_{s,e})^{\alpha_c} \quad (3.18)$$

Effective properties are used in the above equations in which the electrolyte ionic diffusivity, conductivity, diffusional conductivity, and solid phase electric conductivity are defined as:

$$D_e^{eff} = D_e \varepsilon_e^\beta \quad (3.19)$$

$$k^{eff} = k \varepsilon_e^\beta$$

$$k_D^{eff} = \frac{2RTk^{eff}}{F} \left( t_+^0 - 1 \right) \left( 1 + \frac{d \ln f_\pm}{d \ln c_e} \right)$$

$$\sigma^{eff} = \sigma \varepsilon_s^\beta$$

$$a_s = 3\varepsilon_s / r_s$$

$$D_s = D_{s,ref} \exp\left[-E_d / R(1/T - 1/T_{ref})\right]$$

$$k_m = k_{m,ref} \exp\left[-E_r / R(1/T - 1/T_{ref})\right]$$

The above equations use the following variables:

$\sigma^{eff}$  is the effective electric conductivity

$k$  is the electrolyte ionic conductivity

$k_D^{eff}$  is the electrolyte diffusional conductivity

$D_s$  is the diffusion coefficient of Li in solid

$D_e$  is the diffusion coefficient of Li+ in the electrolyte phase

$t_+^0$  is the transference number of lithium ion

$U$  is the open circuit potential of an electrode reaction

$c_{s,\max}$  is the maximum concentration of lithium in solid phase

$c_{s,e}$  is the concentration of lithium at the surface of solid particles

$\alpha_a$  is the charge transfer coefficient at anode

$\alpha_c$  is the charge transfer coefficient at cathode

$a_s$  is the solid/electrolyte interfacial area per unit volume

$F$  is the Faraday constant

$R$  is the universal gas constant

$f_{\pm}$  is the electrolyte activity coefficient

$\beta$  is the Bruggeman porosity exponent

$\varepsilon_e$  is the volume fraction of the electrolyte phase in electrode

$\varepsilon_s$  is the volume fraction of active material in electrode

$\varepsilon_f$  is the volume fraction of the filler material in electrode

$D_{s,\text{ref}}$  is the reference solid diffusion coefficient

$k_{m,\text{ref}}$  is the reference reaction rate constant

$E_d$  is the activation energy that controls the temperature sensitivity of  $D_s$

$E_r$  is the activation energy that controls the temperature sensitivity of  $k_m$

$T_{\text{ref}}$  is the reference temperature, 298 K

The source terms for the potential equations (Equation 3.2 (p. 85)) and the energy equation (Equation 3.1 (p. 85)) is computed as:

$$j_{ECh} = -ai_P \quad (3.20)$$

$$\dot{q}_{ECh} = \frac{i_P (\varphi_+ - \varphi_-) + \int_0^{l_p+l_s+l_n} j^{Li} \left( T_{\text{ref}} \frac{\partial U}{\partial T} - U_{\text{Ref}} \right) dx}{l_p + l_s + l_n} \quad (3.21)$$

where  $a$  is the specific area of the electrode sandwich sheet in the battery,  $l_p$ ,  $l_n$ , and  $l_s$  are the thicknesses of positive electrode, negative electrode, and separator, respectively, and  $i_P$  is the transverse current density

$$i_P = \int_0^{l_p} j^{Li} dx \quad (3.22)$$

### 3.1.5. Coupling Between CFD and Submodels

During computation, an electrochemical submodel is constructed and solved for every CFD cell at every flow iteration (for steady simulations) or timestep (for transient simulations). For the ECM model, this results in solving a set of four algebraic differential equations. For the Newman model, this can involve solving a system of hundreds of algebraic differential equations, depending upon the number of discretization points used in the electrode and particle domain. This adds to a tremendous computational cost of the simulation.

The current transfer rate,  $j$ , is more or less uniform, but current itself is not. As a result, instead of solving the submodel for every computational cell, we can group CFD cells into clusters and solve the submodel only once for each cluster.

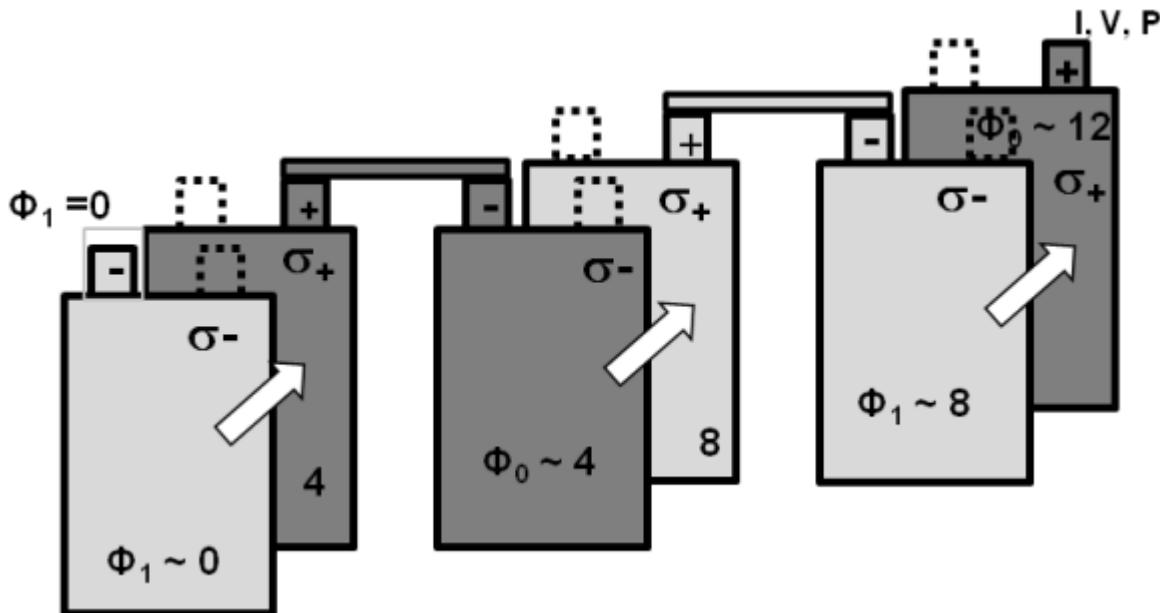
ANSYS Fluent provides a tool for grouping the CFD cells into clusters. For a user-specified number of clusters  $N_x$ ,  $N_y$ , and  $N_z$  in the X, Y, and Z directions respectively, the solver will divide the battery electrochemical domain into  $N_x N_y N_z$  clusters and then obtain the solution for each cluster using the cluster-average values.

### 3.1.6. Battery Pack Simulation

Many applications utilize battery systems in which multiple batteries are connected in series or in parallel. In ANSYS Fluent, the MSMD approach has been extended to simulate battery systems. When simulating a battery pack, ANSYS Fluent uses the continuous zone method in which two potential equations are solved over all the conductive zones continuously. In a single battery case, two potential equations can be interpreted as  $\varphi_+$  and  $\varphi_-$ , where  $\varphi_+$  is the potential field of the positive electrode and  $\varphi_-$  is the potential of the negative electrode. In a multiple cell system, the situation is more complicated as explained below.

[Figure 3.3: Solution Domain for Two Potential Equations in a Battery Pack System \(p. 92\)](#) schematically shows an example in which three battery cells are connected in series.

**Figure 3.3: Solution Domain for Two Potential Equations in a Battery Pack System**



The positive tab of one battery is connected to the next battery's negative tab. As a result, the  $\varphi_+$  of the first battery is about at the same level as the  $\varphi_-$  of the second battery. In other words,  $\varphi_+$  of the first battery and  $\varphi_-$  of the second battery should be solved in the same domain. We still can use only two potential fields to capture the electric field, however, we can no longer interpret them as the potential fields of the positive and negative electrodes.

In [Figure 3.3: Solution Domain for Two Potential Equations in a Battery Pack System \(p. 92\)](#), different shades of gray are used for cell zones to demonstrate the different equations being solved (for the

convenience of bookkeeping, we introduce the "level" of a zone to mark which equation needs to be solved for a given zone):

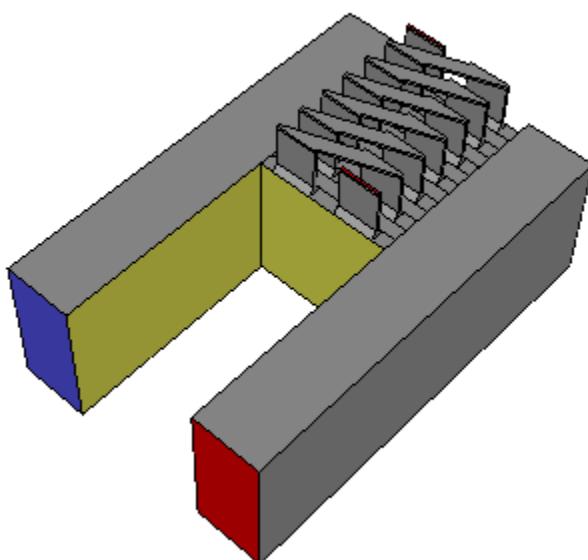
- An active zone (where electro-chemical reactions occur) belongs to two "levels". Both  $\varphi_1$  and  $\varphi_2$  are solved:  $\varphi_1$  is solved for the zones at the odd level and  $\varphi_2$  is solved in the zones at the even level.
- A passive zone (tab or busbar) belongs to only one level, where only one potential equation is solved.

For example, for N batteries connected in series, there will be N+1 levels. Except for the first and the last levels, each level is isolated in the sense that there is no flux across its boundary. Two levels are coupled only through the equations source terms. Two potential equations are solved continuously across all the conductive zones. In a passive zone, however, only one potential equation is solved, and the second equation is excluded by setting the electrical conductivity to zero.

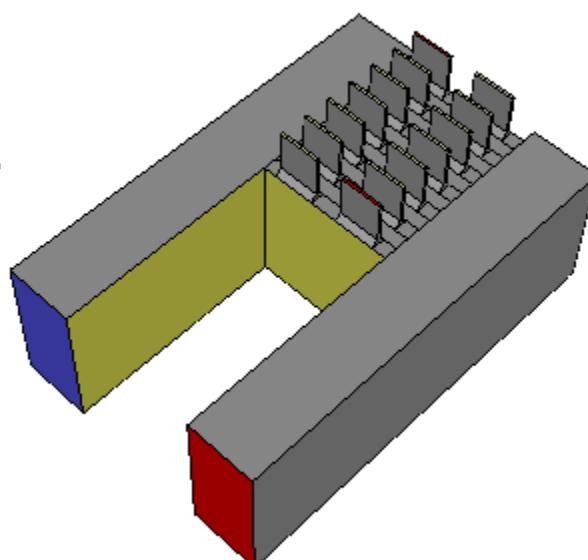
## Battery Connections

Batteries in a pack can be connected through either "real connections" or "virtual connections".

**Real Connections**



**Virtual Connections**



Real connections are busbars that are physically resolved and meshed in the model.

Virtual connections are busbars or battery tab volumes that are not explicitly resolved in the model. Instead, the connection information is provided in a battery connection definition file as described in [Specifying Electric Contacts \(p. 115\)](#). The solver reads this information and sets the electric boundary conditions for each individual battery through the virtual connections.

Using virtual connections saves the time and effort of meshing busbar and tab volumes, which are usually very thin and difficult to mesh. The disadvantage is that electric resistance and Joule heating are not considered in these unresolved volumes.

You can select the method that best suits your needs in your simulation. Refer to [Specifying Electric Contacts \(p. 115\)](#) for the details on using the battery virtual connections in a battery pack.

### 3.1.7. Reduced Order Solution Method (ROM)

The MSMD method needs to solve two potential equations (Equation 3.2 (p. 85)). Although the equations do not have transient terms, they are solved repeatedly during each time step in a transient fashion due to the time-varying electric load condition and the change of state of a battery charge.

However, under the following two conditions, the solution of these two potential equations can be easily found without solving them repeatedly:

- Electric conductivity does not depend on temperature
- Transfer current density is uniform over a battery's active zone

A careful examination of Equation 3.2 (p. 85) and their boundary conditions reveals that the two equations become homogeneous linear partial differential equations under those two conditions. Instead of solving them at each time step, a scaling procedure can be used, so that the two potential fields under any electric load condition can be obtained from a reference potential field. The reference potential field can be pre-calculated at a reference load condition. By using this method, the need to solve the two potential equations at each CFD time step is eliminated entirely. The cost of a general electrochemical-thermal coupled simulation is reduced to that of a pure thermal simulation. This can save a significant amount of computational time in a battery simulation.

### 3.1.8. External and Internal Electric Short-Circuit Treatment

The ANSYS Fluent battery model can handle both external and internal short-circuits.

#### ***External Short-Circuit***

In the case of an external short-circuit, battery's positive and negative tabs are connected directly with no electric load. In ANSYS Fluent, an electric load boundary condition for the external resistance enables you to directly specify the total external electric resistance,  $R_{load}$ . The battery model solver will guarantee that the ratio of the battery's tab voltage,  $V_{tab}$ , to the tab current,  $I_{tab}$ , equals the specified resistance value:

$$R_{load} = V_{tab} / I_{tab}$$

#### ***Internal Short-Circuit***

Under normal battery operation, the battery's positive and negative electrodes are divided by a separator, usually a thin polymer material that prevents electrons from traveling directly from the negative electrode to the positive electrode. In the case of an internal short-circuit, which could be a result of a nail penetration or a crash accident, the separator is ruptured in a localized area. Besides providing the normal tab current, the battery produces a secondary electric current from electrochemical reactions that is wasted through the shorted area.

The short-circuit intensity can be described by a fundamental variable-volumetric contact resistance, which could be computed as  $r_c/a$ , where  $r_c$  is the contact resistance of the electrode sheet in the unit of  $\Omega\text{m}^2$ , and  $a$  is the specific area of the electrode sheet in the battery volume in units of  $1/\text{m}$ . The transfer short current density,  $j_{short}$  in Equation 3.2 (p. 85), and extra heat generated due to the short are computed as:

$$\begin{aligned} j_{short} &= a \dot{j}_{short} = a(\phi_+ - \phi_-) / r_c \\ \dot{q}_{short} &= a(\phi_+ - \phi_-)^2 / r_c \end{aligned} \quad (3.23)$$

The above formulation is general enough to cover non-short cases where  $r_c/a$  tends to infinity. Contact resistance could be a function of time and location, therefore allowing the dynamic evolution of the changes in the size of the shorted region and the intensity of the short-circuit current.

For information on how to use and set up this feature, see [Specifying External and Internal Short-Circuit Resistances \(p. 118\)](#).

### 3.1.9. Thermal Abuse Model

Safety has become an important issue in battery design. Under abuse conditions, thermal runaway may occur in a battery. Increased temperature could trigger thermal runaway reactions. Excessive heat released as a result of such conditions could damage a battery cell, or even cause fire or explosion. Thermal runaway reactions are very complicated and material-specific. For modeling thermal runaway reactions and simulating a battery thermal behavior under thermal abuses, ANSYS Fluent offers two semi-empirical models:

- One-Equation Model
- Four-Equation Model

For details on how to set up and use these model, see [Specifying Advanced Option \(p. 116\)](#).

#### One-Equation Model

In the one-equation model, thermal runaway reactions are lumped together as one reaction that can be described by the following kinetics equation:

$$\frac{d\alpha}{dt} = A \cdot \exp(-E/RT) \alpha^m (1-\alpha)^n \quad (3.24)$$

where:

$\alpha$  = degree of conversion

$A$  = pre-exponential factor of the reaction ( $s^{-1}$ )

$E$  = activation energy of the reaction (J/mol)

$R$  = universal gas constant

$T$  = temperature

$m$  and  $n$  = reaction order parameters

[Equation 3.24 \(p. 95\)](#) is solved to keep track of the progress of the thermal runaway reaction.  $\alpha$  changes from 0 to 1, with 0 denoting "no reaction" and 1 denoting "completion of reaction".

With  $\alpha$  computed from [Equation 3.24 \(p. 95\)](#), the heat generation rate due to thermal runaway can be calculated as:

$$\dot{q}_{abuse} = HW \cdot \left| \frac{d\alpha}{dt} \right| \quad (3.25)$$

where  $HW$  is the specific heat release ( $J/m^3$ ).

## Four-Equation Model

In the four-equation model, thermal runaway reactions are put into the following four categories [7] (p. 131):

- solid electrolyte interface (SEI) decomposition reactions
- negative electrode-electrolyte reactions
- positive electrode-electrolyte reactions
- electrolyte decomposition reactions

To keep track of the progress of each type of reactions, a lumped Arrhenius kinetics rate is used:

$$\frac{dc_{sei}}{dt} = -A_{sei} \exp\left(-\frac{E_{sei}}{RT}\right) c_{sei}^{m_{sei}} \quad (3.26)$$

$$\frac{dc_{ne}}{dt} = -A_{ne} \exp\left(-\frac{t_{sei}}{t_{sei,ref}}\right) \exp\left(-\frac{E_{ne}}{RT}\right) c_{ne}^{m_{ne}} \quad (3.27)$$

$$\frac{d\alpha}{dt} = A_{pe} \exp\left(-\frac{E_{pe}}{RT}\right) \alpha^{m_{pe,1}} (1-\alpha)^{m_{pe,2}} \quad (3.28)$$

$$\frac{dc_e}{dt} = -A_e \exp\left(-\frac{E_e}{RT}\right) c_e^m \quad (3.29)$$

where:

$A$ ,  $E$ , and various  $m$  are reaction kinetics parameters, representing pre-exponential factor, activation energy, and reaction order, respectively

subscripts  $sei$ ,  $ne$ ,  $pe$ , and  $e$  denote parameters associated with SEI decomposition reaction, negative electrode-electrolyte reaction, positive electrode-electrolyte reaction, and electrolyte decomposition reaction, respectively

$c_{sei}$ ,  $c_{ne}$ ,  $\alpha$ , and  $c_e$  are dimensionless variables that can be interpreted as the fraction of the remaining reactant associated to each type of reactions in the medium as reactions proceed

$t_{sei}$  is the dimensionless measure of a SEI layer thickness

$t_{sei,ref}$  is the reference SEI layer thickness

$T$  is the temperature

$R$  is the universal gas constant

The values of  $c_{sei}$ ,  $c_{ne}$ , and  $c_e$  change from 1 (representing the state when nothing has reacted yet) to 0 (denoting the state with reactants being completely consumed).  $\alpha$ , on the other hand, varies from 0 to 1, as in the One-Equation model.

$t_{sei}$  can be computed as:

$$t_{sei} = t_{sei,0} + (c_{neg,0} - c_{neg})$$

where  $t_{sei,0}$  and  $c_{neg,0}$  are two initial values. Note that the SEI layer is growing as the negative electrode-electrolyte reaction progresses.

The total heat generation due to the thermal runaway reactions (in units of W/m<sup>3</sup>) can be computed as:

$$\dot{q}_{abuse} = H_{sei}W_{sei}\left|\frac{dc_{sei}}{dt}\right| + H_{ne}W_{ne}\left|\frac{dc_{ne}}{dt}\right| + H_{pe}W_{pe}\left|\frac{d\alpha}{dt}\right| + H_eW_e\left|\frac{dc_e}{dt}\right| \quad (3.30)$$

where  $H$  is the heat of reaction (J/kg), and  $W$  (kg/m<sup>3</sup>) is density of reactants in the medium.

## 3.2. Using the Dual-Potential MSMD Battery Model

This section describes how to use the MSMD modeling capability in ANSYS Fluent. Only the procedural steps related to battery modeling are shown here. For information about inputs related to other models used in conjunction with the battery model, see the appropriate sections for those models in the ANSYS Fluent User's Guide. Refer to [Dual-Potential MSMD Battery Model Theory \(p. 85\)](#) for information about the theory behind the MSMD modeling approach.

Information about using the model is presented in the following sections:

- [3.2.1. Limitations](#)
- [3.2.2. Geometry Definition for the Dual-Potential MSMD Battery Model](#)
- [3.2.3. Loading the Dual-Potential MSMD Battery Module](#)
- [3.2.4. Setting up the Dual-Potential MSMD Battery Model](#)
- [3.2.5. Initializing the Battery Model](#)
- [3.2.6. Modifying Material Properties](#)
- [3.2.7. Solution Controls for the Dual-Potential MSMD Battery Model](#)
- [3.2.8. Postprocessing the Dual-Potential MSMD Battery Model](#)
- [3.2.9. User-Accessible Functions](#)
- [3.2.10. Using the Dual-Potential MSMD Battery Model Text User Interface](#)

### 3.2.1. Limitations

Note the following limitations when using the Dual-Potential MSMD Battery model:

- For battery pack simulations, all batteries in the pack must be identical initially, having the same battery type with the same initial conditions.
- For battery pack simulations, all batteries in the pack must have the nPmS connection pattern; that is,  $n$  batteries connected in parallel, and then  $m$  such units connected in series.
- A conductive zone must be continuous. Disconnected zones cannot be grouped as one entity.
- System positive and negative tabs must be continuous surfaces. Disconnected faces cannot be grouped together as one entity.
- The positive and negative tabs must be connected by either the real tab/busbar conductive volumes or the virtual connections defined through a file.
- A battery tab cannot be directly contact another battery tab. They must be connected by busbars.

### 3.2.2. Geometry Definition for the Dual-Potential MSMD Battery Model

In the MSMD framework, the negative and positive current collectors, anode, and cathode are not resolved, and the battery is regarded as a homogeneous body. The computational mesh does not have to be adjusted to all layers present in the battery domain. The following zones have to be identified when creating the battery mesh:

- Cell
- Tab
- Busbar (for battery pack cases with real battery connections)

When using the real battery connections you need to create busbar zones and specify them under **Busbar Components** (see [Specifying Conductive Zones \(p. 114\)](#) for details). Since both tab zones and busbar zones are passive zones, you can group them all under busbar for simplicity. However, busbars sometimes may not be assigned to the tab zone category because a battery tab zone cannot directly connect to another battery tab zone.

When using the virtual battery connections you do not need to create busbar zones. You only need to create tab zones (if you have tab volumes) and specify them under **Tab Components** (see [Specifying Conductive Zones \(p. 114\)](#) for details).

### 3.2.3. Loading the Dual-Potential MSMD Battery Module

The battery module is loaded into ANSYS Fluent through the text user interface (TUI). The module can be loaded only when a valid ANSYS Fluent case file has been set or read. The text command to load the module is

```
define → models → addon-module
```

A list of ANSYS Fluent add-on modules is displayed:

```
Fluent Addon Modules:  
0. none  
1. MHD Model  
2. Fiber Model  
3. Fuel Cell and Electrolysis Model  
4. SOFC Model with Unresolved Electrolyte  
5. Population Balance Model  
6. Adjoint Solver  
7. Single-Potential Battery Model  
8. Dual-Potential MSMD Battery Model  
9. PEM Fuel Cell Model  
Enter Module Number: [0] 8
```

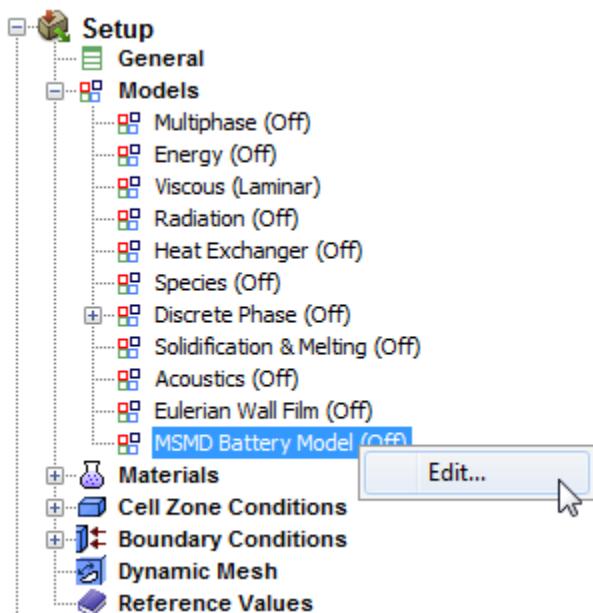
Select the dual-potential MSMD battery model by entering the module number 8. During the loading process, a scheme library containing the graphical and text user interface, and a UDF library containing a set of user-defined functions (UDFs) for the battery module are loaded into ANSYS Fluent. This is reported to the console. The UDF library also becomes visible as a new entry in the **UDF Library Manager** dialog box. The basic setup of the battery model is performed automatically when the battery module is successfully loaded.

### 3.2.4. Setting up the Dual-Potential MSMD Battery Model

Once the module has been loaded, you can enable the Dual-Potential MSMD Battery Model, set battery model parameters and assign properties to the relevant regions in your battery.

To enable the MSMD battery model: In the tree under the **Models** branch, right-click **MSMD Battery Model** and click **Edit...** in the menu that opens.



**Figure 3.4: The MSMD Battery Model Option in the Tree**

In the **MSMD Battery Model** dialog box that opens, select the **Enable Battery Model** check box to activate the model.



The **MSMD Battery Model** dialog box expands to reveal additional model options and solution controls.

The inputs for the battery model are entered using the following tabs:

#### **Model Options**

contains the general model settings to define problems using the battery model.

#### **Model Parameters**

is where you specify the parameters to be used for solving the model equations.

#### **Conductive Zones**

allows you to select zones for the active, tab and busbar components.

#### **Electric Contacts**

allows you to define the contact surface and external connectors.

## **Advanced Option**

allows you to enable a thermal abuse model and specify its parameters.

---

### **Note**

When setting up the model, you can click the **Apply** button to save your input values and your choices for your model. To restore the last saved settings for all tabs, click the **Reset** button.

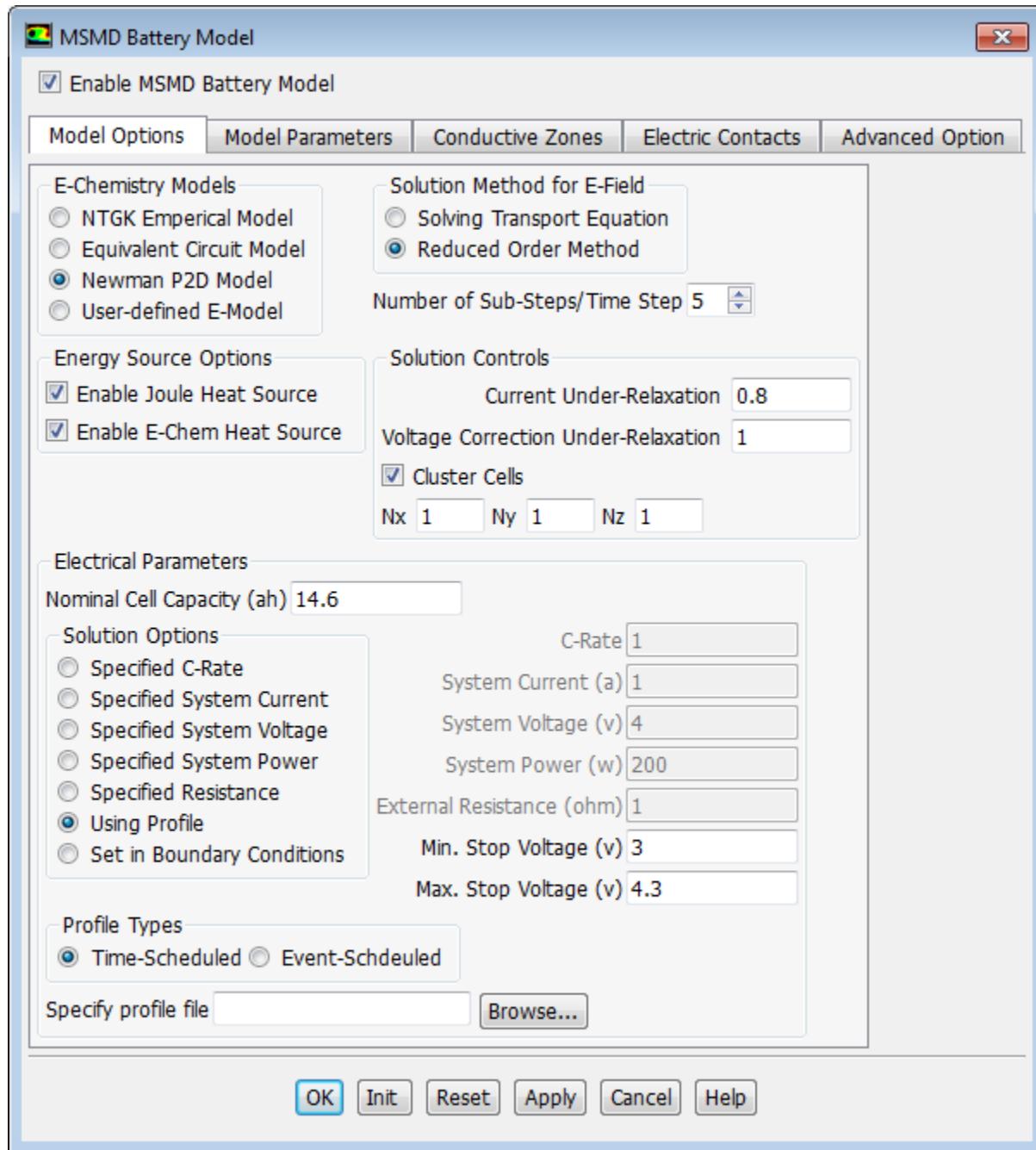
---

For additional information, see the following sections:

- [3.2.4.1. Specifying Battery Model Options](#)
- [3.2.4.2. Specifying Battery Model Parameters](#)
- [3.2.4.3. Specifying Conductive Zones](#)
- [3.2.4.4. Specifying Electric Contacts](#)
- [3.2.4.5. Specifying Advanced Option](#)
- [3.2.4.6. Specifying External and Internal Short-Circuit Resistances](#)

### **3.2.4.1. Specifying Battery Model Options**

In the **Model Options** tab, you can select the electrochemical battery submodel and set various model options, solution controls, and electrical parameters.

**Figure 3.5: The MSMD Battery Model Dialog Box (Model Options Tab)**

The following submodels are available in ANSYS Fluent under the **E-Chemistry Models** group box:

#### NTGK Empirical Model

is an empirical model. This is the simplest and the least expensive model. It also requires the fewest model inputs. The model uses the same NTGK method as that used by the Single-Potential Empirical model. However, in this case, it solves two potential equations continuously instead of one potential equation with a jump condition at the separator over all conductive zones. The model cannot accurately capture the battery's dynamic (inertial) response.

#### Equivalent Circuit Model

is a semi-empirical model. It has limited capability for predicting the battery dynamic behavior.

**Newman P2D Model**

is the most comprehensive physics-based battery model. It is computationally expensive, and it requires extensive user input of battery-related fundamental information.

**User-defined E-Model**

is the user-specified electrochemical submodel. To be able to use this option in the Fluent MSMD method framework, you must provide the `void user_defined_echem_model()` function in the customizable `cae_user.c` source code file.

For **Solution Method for E-Field**, if you select:

- **Solving Transport Equation**, two potential equations will be solved during each iteration in the simulation.
- **Reduced Order Method**, the solution will be obtained (faster) using the reduced order method (ROM). You can use this option if the two conditions listed in [Reduced Order Solution Method \(ROM\) \(p. 94\)](#) are met. See [Reduced Order Solution Method \(ROM\) \(p. 94\)](#) for details.

Note that, prior to using the ROM, you must first run a simulation with the **Solving Transport Equation** option for at least three time steps. ANSYS Fluent will use these results as reference potential fields for the ROM method.

Once you have selected **Reduced Order Method**, you can set **Number of Sub-Steps/Time Step**. Using this value, you can specify the ratio of the time steps for thermal and electro-chemistry calculations. For example, if the **Number of Sub-Stepes/Time Step** is set to 10, the solver will use one tenth of the CFD time step for the electro-chemistry calculation.

---

**Important**

If internal short circuit occurs in a battery, the **Reduced Order Method** cannot be used.

---

For **Energy Source Options**, you can:

- **Enable Joule Heat Source** in order to include the Joule Heating source in the thermal energy equation [Equation 3.1 \(p. 85\)](#). (enabled by default)
- **Enable E-Chem Heat Source** in order to include the heat source due to electrochemistry in the thermal energy equation [Equation 3.1 \(p. 85\)](#). (enabled by default)

For **Solution Controls**, you can:

- Set **Current Under-Relaxation Factor**. The volumetric current density used in the two potential equations are under-relaxed according to this factor. The default value of 0.8 may be sufficient for most cases. The values in the range of 0.8–1.0 are recommended.
- Set **Voltage Correction Under-Relaxation Factor**. The default value of 1.0 is acceptable for most cases. You should reduce this value only if the convergence difficulty occurs during the solution of the potential equations.
- (Equivalent Circuit Model and Newman P2D Model) Select the **Cluster Cells** check box to enable cell clustering and specify the number of clusters **N<sub>x</sub>**, **N<sub>y</sub>**, and **N<sub>z</sub>** in X, Y, and Z directions respectively. The solver will divide the battery domain into **N<sub>x</sub> × N<sub>y</sub> × N<sub>z</sub>** clusters and solve submodel equations for each cluster using

cluster-average information. The default values for  $N_x$ ,  $N_y$ , and  $N_z$  are 1, 1, and 1, respectively. For the initial run, the default settings are recommended.

For **Electrical Parameters**, you can specify the following operation conditions:

- Set the **Nominal Cell Capacity** (the capacity of the battery cell).

- For the **Solution Options**, if you select:

- **Specified C-Rate**, you can set a value for the following:

→ **Discharge C-Rate** (the hourly rate at which a battery is discharged; a positive value means discharging C-rate and a negative value means charging C-rate). In this case, the total current at the cathode tabs is fixed as the product of C-Rate and Nominal Capacity, while the electrical potential is anchored at zero on the anode tabs.

→ **Min. Stop Voltage** and **Max. Stop Voltage**. The simulation will automatically stop once the battery cell voltage reaches the minimum or maximum value. For a battery pack, the battery-average cell voltage is used.

- **Specified System Current**, you can set the following:

→ **System Current** (applied at the cathode tabs). A positive value means discharging current and a negative value means charging current.

→ **Min. Stop Voltage** and **Max. Stop Voltage**. (Same as for **Specified C-rate**.)

- **Specified System Voltage**, you can set a value for the **System Voltage** (applied to the cathode tab; the anode tab has a voltage of 0 V).

- **Specified System Power**, you can specify the following:

→ **System Power**. If you specify the total power output from your battery, Fluent solver will ensure that the product of the system current and voltage equals the system power. A positive value means discharging power output and a negative value means charging power input.

→ **Min. Stop Voltage** and **Max. Stop Voltage**. (Same as for **Specified C-rate**.)

- **Specified Resistance**, you can specify the external electric resistance in an external short-circuit simulation.

→ **External Resistance**. If you specify the external electric resistance in Ohm, the Fluent solver will ensure that the ratio of the system voltage to the system current equals the specified external resistance.

- **Set in Boundary Conditions**, you can set the UDS boundary conditions directly, for example, the voltage value or the current value (specified flux), using the **Boundary Conditions** task page in ANSYS Fluent for the specific face zone.

- **Using Profile**, you can select **Time-Scheduled** or **Event-Scheduled** under **Profile Type** and provide time-dependent or event-dependent input (respectively) for the C-rate, current, voltage, or power. The time-scheduled and event-scheduled profiles enable you to change the electric load type and electric value in the simulation on the fly.

Once the profile type is selected, you can browse to the ASCII text file with either the .dat or .txt file extension that you generated beforehand.

The inputs for the **Time-Scheduled** and **Event-Scheduled** profiles are described below.

→ The **Time-Scheduled** profile file must be organized in three columns:

1. Time
2. Electric load value
3. Electric load type

The electric load type is defined by an integer number:

0: C-rate

1: Current

2: Voltage

3: Power

4: External electric resistance

For example, the below time-scheduled profile:

```
0      15    1
500    15    1
500.1   60    3
1000   60    3
1000.1  -1    0
1500   -1    0
```

specifies the following battery charging scenario:

1. The battery is discharged at 15 A for the first 500 seconds.
  2. The battery discharge continues at 60 W for the next 500 seconds.
  3. The battery is charged at a constant 1 C-rate for the last 500 seconds.
- 

### Note

- For current, power, or C-rate, a positive value means discharging and a negative value means charging.
  - The unit of the variables in the input file is always SI.
- 

ANSYS Fluent solver uses linear interpolation of your time-scheduled profile data in the problem simulation. If you want to capture a sudden change in the profile accurately, use smaller time gaps to describe the profile.

→ The **Event-Scheduled** profile file must be organized in the following five columns:

1. Electric load type

Just as in the time-scheduled profile, the electric load type is defined by an integer number:

- 0: C-rate**
- 1: Current**
- 2: Voltage**
- 3: Power**
- 4: External electric resistance**

2. Electric load value
3. Forwarding condition

The forwarding condition (FC) is defined by an integer number:

- 1: Time > FC\_Value**
  - 2: I < FC\_Value (discharging current minimum)**
  - 3: I > FC\_Value (charging current maximum)**
  - 4: V < FC\_Value (system voltage minimum)**
  - 5: V > FC\_Value (system voltage maximum)**
  - 6: P < FC\_Value (discharging power minimum)**
  - 7: P > FC\_Value (charging power maximum)**
4. Forwarding condition value
  5. Stop flag

The stop flag is an integer value of 0 or 1, where 0 specifies that the run should continue to the next event entry line, and 1 specifies that the run should end.

For example, the below time-scheduled profile:

0	1	4	3.5	0
1	-15	5	4.1	0
2	4.1	3	-1	1

specifies the following scenario:

1. The battery is discharged at 1 C rate until the voltage reaches 3.5 V.
2. The battery is charged at constant current 15 A until the voltage reaches 4.1 V.

3. The battery keeps charging at 4.1 V until the current is greater -1 A. The simulation stops after reaching this point.
- 

#### Note

- For current, power, or C-rate, a positive value means discharging and a negative value means charging.
  - The unit of the variables in the input file is always SI.
- 

Any time-scheduled profile can be rewritten in an event-scheduled profile format, but not vice-versa. The example shown in the time-scheduled profile can be redefined in an event-scheduled profile format as follows.

```
1      15      1      500      0
3      60      1     1000      0
0     -1      1     1500      1
```

Both profile types can be read into ANSYS Fluent. You can choose the most convenient one to use in your specific application.

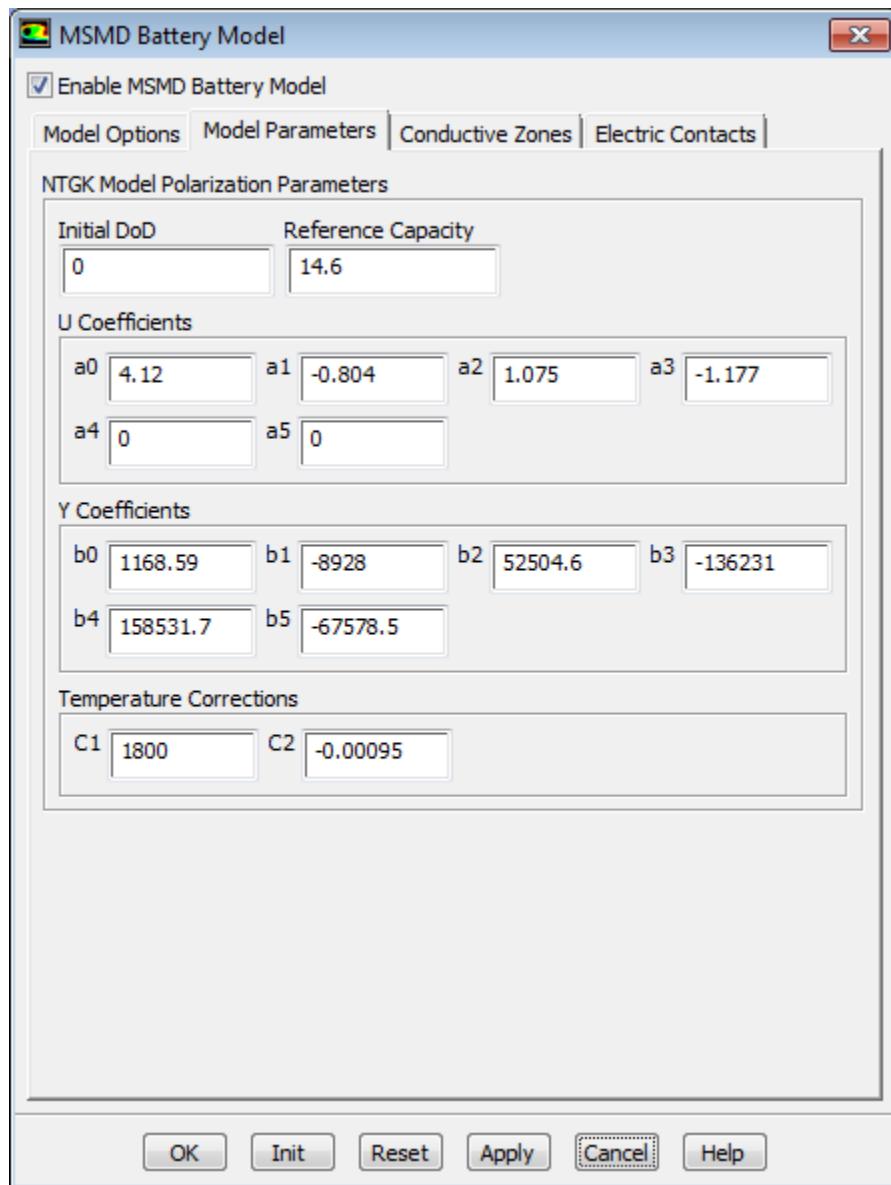
### **3.2.4.2. Specifying Battery Model Parameters**

The **Model Parameters** tab of the **MSMD Battery Model** dialog box allows you to specify the submodel-specific parameters to be used for solving the submodel equations. The **Model Parameters** tab contains only the entry fields for parameters related to the electrochemical submodel you have chosen in the **Model Option** tab. A detailed description of inputs for each submodel is given in the sections that follow.

- [3.2.4.2.1. Inputs for the NTGK Empirical Model](#)
- [3.2.4.2.2. Inputs for the Equivalent Circuit Model](#)
- [3.2.4.2.3. Inputs for the Newman's P2D Model](#)
- [3.2.4.2.4. Input for the User-Defined E-Model](#)

### 3.2.4.2.1. Inputs for the NTGK Empirical Model

**Figure 3.6: Model Parameters Tab—NTGK Model**



For the **NTGK Empirical Model**, you can specify polynomial coefficients for  $U$ ,  $Y$ , and temperature correction in [Equation 3.5 \(p. 87\)](#).

#### Note

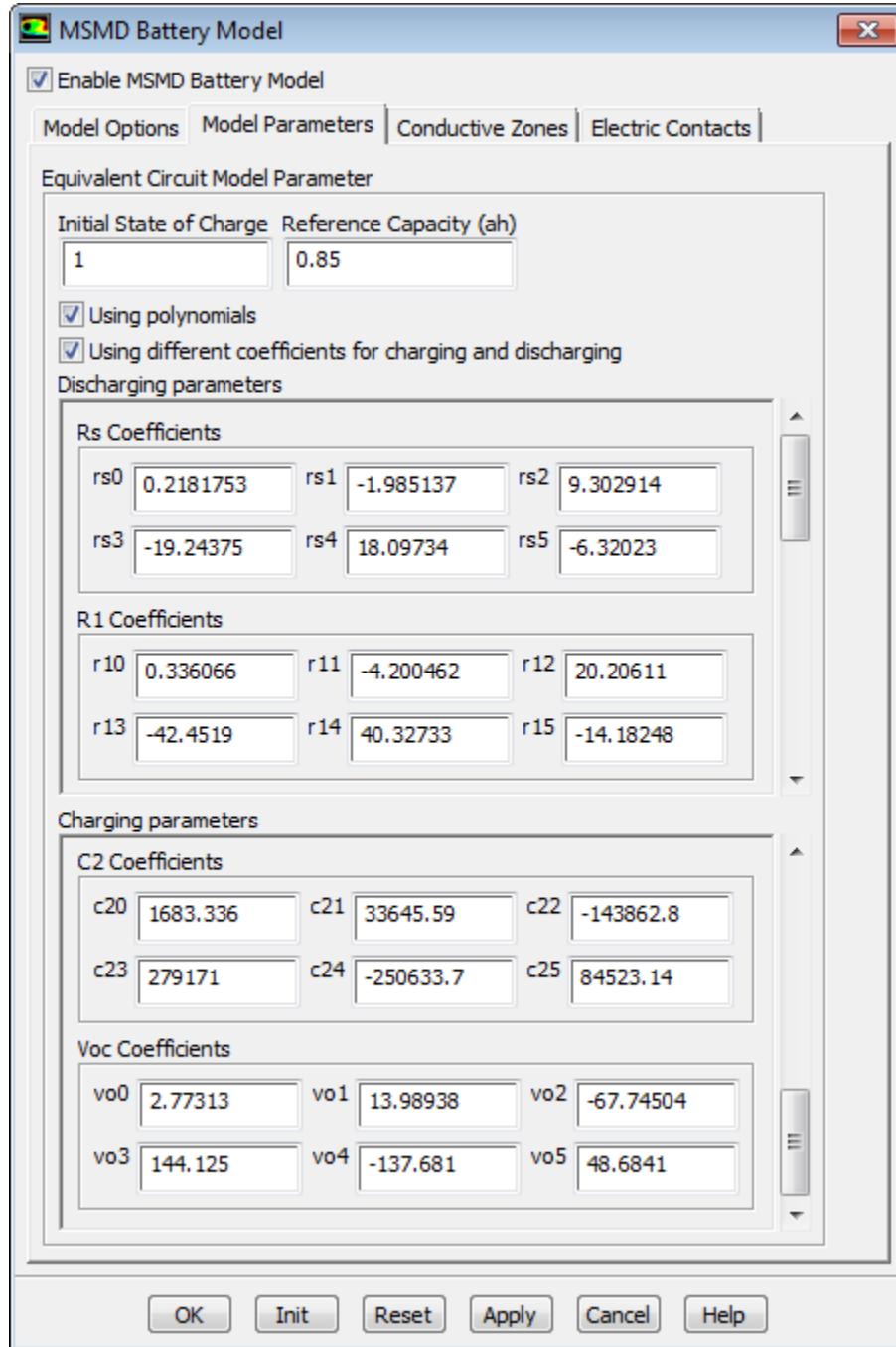
- The polynomial coefficients for the **NTGK Model Polarization Parameters** are based on battery cell polarization test curves. Obtaining polynomial coefficients (other than the default coefficients) can be dependant on your battery configuration and material properties. The default coefficients are taken from Kim's paper [\[9\] \(p. 131\)](#). If you model a battery having a different design, you must provide a different set of battery parameters.
- ANSYS Fluent provides the parameter estimation tool for computing the coefficients of the  $U$  and  $Y$  polynomial functions from the battery's testing data—constant discharging curves. The

tool is available via the text-user-interface (TUI). For details, see [Using the Dual-Potential MSMD Battery Model Text User Interface \(p. 124\)](#)

You can also set values for **Initial DoD** and **Reference Capacity**. The default value of 0 for **Initial DoD** indicates the fully-charged state of the battery cell.

### 3.2.4.2.2. Inputs for the Equivalent Circuit Model

**Figure 3.7: Model Parameters Tab—Equivalent Circuit Model**



For the **Equivalent Circuit Model**, you can specify the function coefficients for the resistors' resistances ( $R_s$ ,  $R_1$ , and  $R_2$ ), capacitors' capacitances ( $C_1$  and  $C_2$ ), and the open circuit voltage ( $V_{oc}$ ) to be used in [Equation 3.9 \(p. 88\)](#).

If you want to specify the ECM model functions in the polynomial form, select **Using polynomials** and enter parameters for [Equation 3.8 \(p. 88\)](#).

In addition, you can turn on the **Using different coefficient for charging and discharging** option and specify the coefficients for **Charging parameters**.

You can also set values for **Initial State of Charge** and **Reference Capacity**. The default value of 1 for **Initial State of Charge** indicates the fully-charged state of the battery cell.

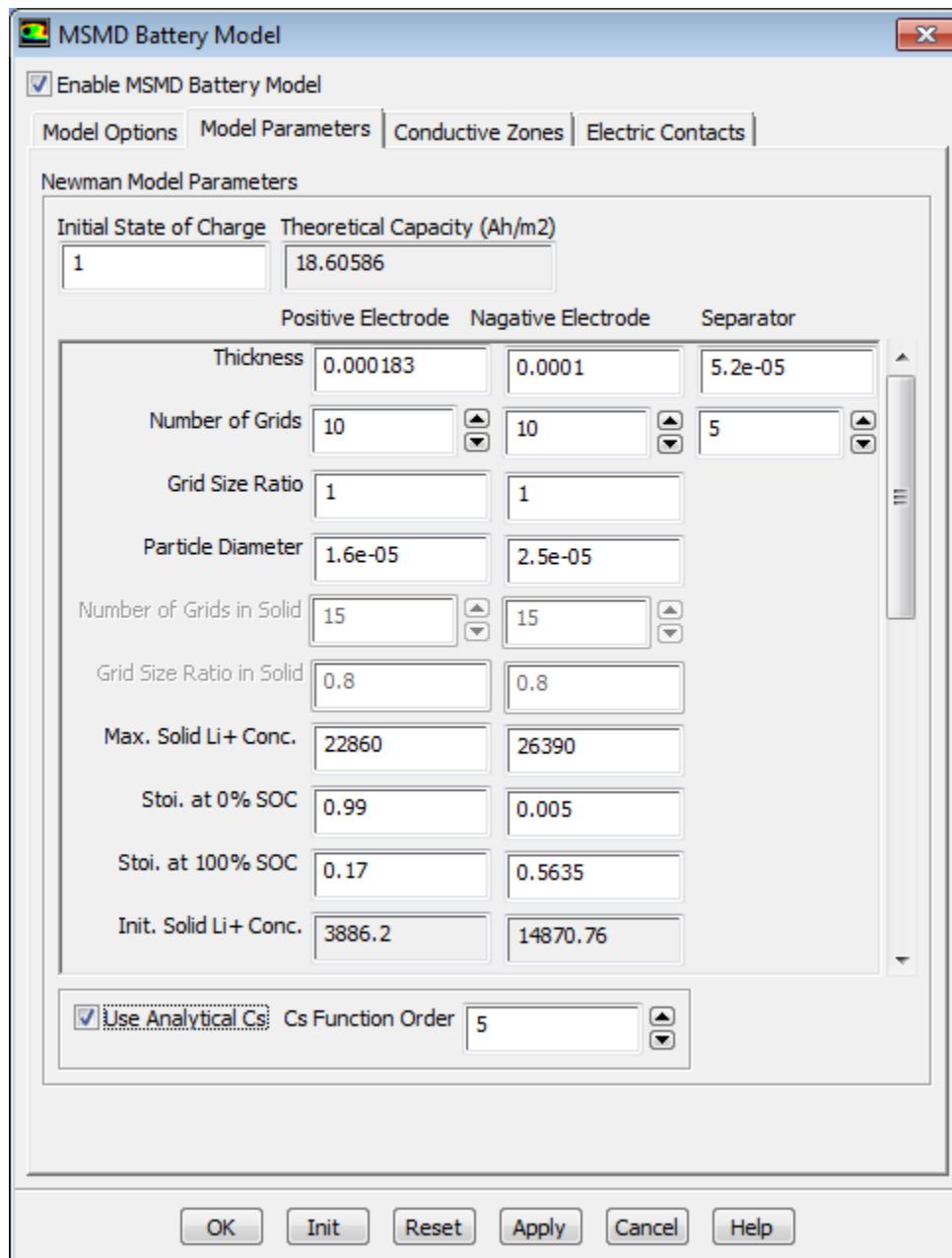
---

### Note

- The polynomial coefficients for the **Equivalent Circuit Model** are battery specific and need to be specified before conducting a simulation. The default coefficients are taken from Chen [5] (p. 131). If you model a battery of different design, you should specify a different set of battery parameters.
  - ANSYS Fluent provides the parameter estimation tool for computing the polynomial coefficients of all the ECM parameters from the battery's testing data—typically, the HPPC data. The tool is available via the text-user-interface (TUI). For details, see [Using the Dual-Potential MSMD Battery Model Text User Interface \(p. 124\)](#).
-

### 3.2.4.2.3. Inputs for the Newman's P2D Model

**Figure 3.8: Model Parameters Tab—Newman's P2D Model**



The following parameters are available under **Model Parameters** for the Newman's P2D model:

#### Thickness

are  $l_p$ ,  $l_n$ , and  $l_s$  in [Equation 3.21 \(p. 91\)](#).

#### Number of Grids

is a number of grid points used to discretize the positive electrode, negative electrode, and separator zones.

#### Grid Size Ratio

is used to create a biased mesh. The default value of 1 corresponds to a uniform mesh.

**Particle Diameter**

is the particle diameter of the active material.

**Number of Grids in Solid**

is the number of grid points used to discretize the sphere of particles.

**Grid Size Ratio in Solid**

is used to create a biased mesh for the solid phase. The default value of 1 corresponds to a uniform mesh.

**Max. Solid Li+ Conc**

is the maximum concentration of Lithium in the solid phase (used in [Equation 3.18 \(p. 90\)](#)).

**Stoi. at 0% SOC**

is the stoichiometry at 0% SOC that is used to compute theoretical capacity and initial lithium concentration in the electrode.

**Stoi. at 100% SOC**

is the stoichiometry at 100% SOC that is used to compute theoretical capacity and initial lithium concentration in the electrode.

**Init. Solid Li+Conc.**

is the initial lithium concentration in the electrode.

**Init. Electrolyte Li+Conc.**

is the initial lithium concentration in the electrolyte phase.

**Volume Fraction**

is the volume fraction of the active material in the electrode,  $\varepsilon_s$  in [Equation 3.19 \(p. 90\)](#).

**Filler Fraction**

is the volume fraction of the filler material in the electrode,  $\varepsilon_f$  in [Equation 3.19 \(p. 90\)](#).

**Ref. Diffusivity**

is the diffusion coefficient of lithium ion in the electrode at reference temperature  $T_{ref}=25^\circ\text{C}$ ,  $D_{s,ref}$  in [Equation 3.19 \(p. 90\)](#).

**Activation Energy E\_d**

is  $E_d$  in [Equation 3.19 \(p. 90\)](#).

**Tortuosity**

is the Bruggeman porosity exponent,  $\beta$  in [Equation 3.19 \(p. 90\)](#).

**Conductivity**

is the electric conductivity,  $\sigma$  in [Equation 3.19 \(p. 90\)](#).

**Ref. Rate Constant**

is the rate constant at reference temperature  $T_{Ref}=25^\circ\text{C}$ ,  $k_{m,ref}$  in [Equation 3.19 \(p. 90\)](#).

**Activation Energy E\_r**

is  $E_r$  in [Equation 3.19 \(p. 90\)](#).

**Trans. Coef. a**

is the charge transfer coefficient at anode,  $\alpha_a$  in [Equation 3.18 \(p. 90\)](#).

**Trans Coef. c**

is the charge transfer coefficient at cathode,  $\alpha_c$  in [Equation 3.18 \(p. 90\)](#).

**Electrolyte Diffusivity**

is the diffusion coefficient of Li<sup>+</sup> in the electrolyte phase,  $D_e$  in [Equation 3.18 \(p. 90\)](#).

**t+ Factor**

is the transference number of lithium ion,  $t_+$  in [Equation 3.19 \(p. 90\)](#).

You can also set values for **Initial State of Charge**. The default value of 1 for **Initial State of Charge** indicates the fully-charged state of the battery cell.

**Theoretical Capacity** is computed from the Newman input parameters automatically and reported in the **MSMD Battery Model** dialog box.

By default, ANSYS Fluent discretizes the lithium conservation equation and solves it numerically. ANSYS Fluent also offers an option to use the reduced order method to solve the lithium diffusion equation in solid [\[4\] \(p. 131\)](#). To use this method, select the **Use Analytical Cs** check box and enter the value for the **Cs Function Order**.

---

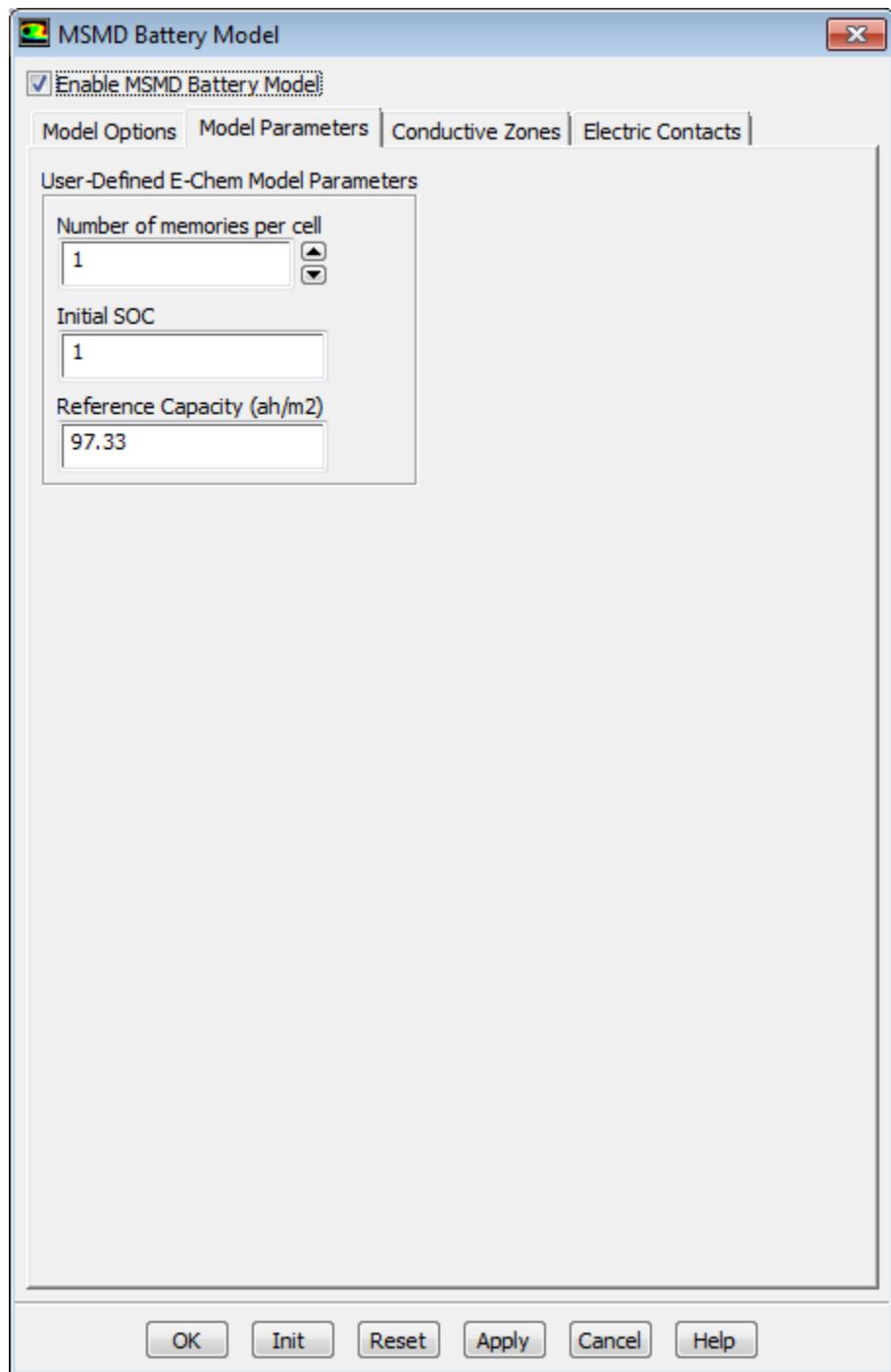
**Note**

Model parameters are battery specific. The default values are taken from Cai & White [\[4\] \(p. 131\)](#). To simulate a battery of different design, you should provide a user-specified set of battery parameters.

---

### 3.2.4.2.4. Input for the User-Defined E-Model

Figure 3.9: Model Parameters Tab—User-Defined E-Model



The following parameters are available under **User-Defined E-Chem Model Parameters**:

**Number of memories per cell**

is the number of submodel parameters that need to be saved per computational cell during a simulation.

**Initial SOC**

is the value of the battery initial state of charge.

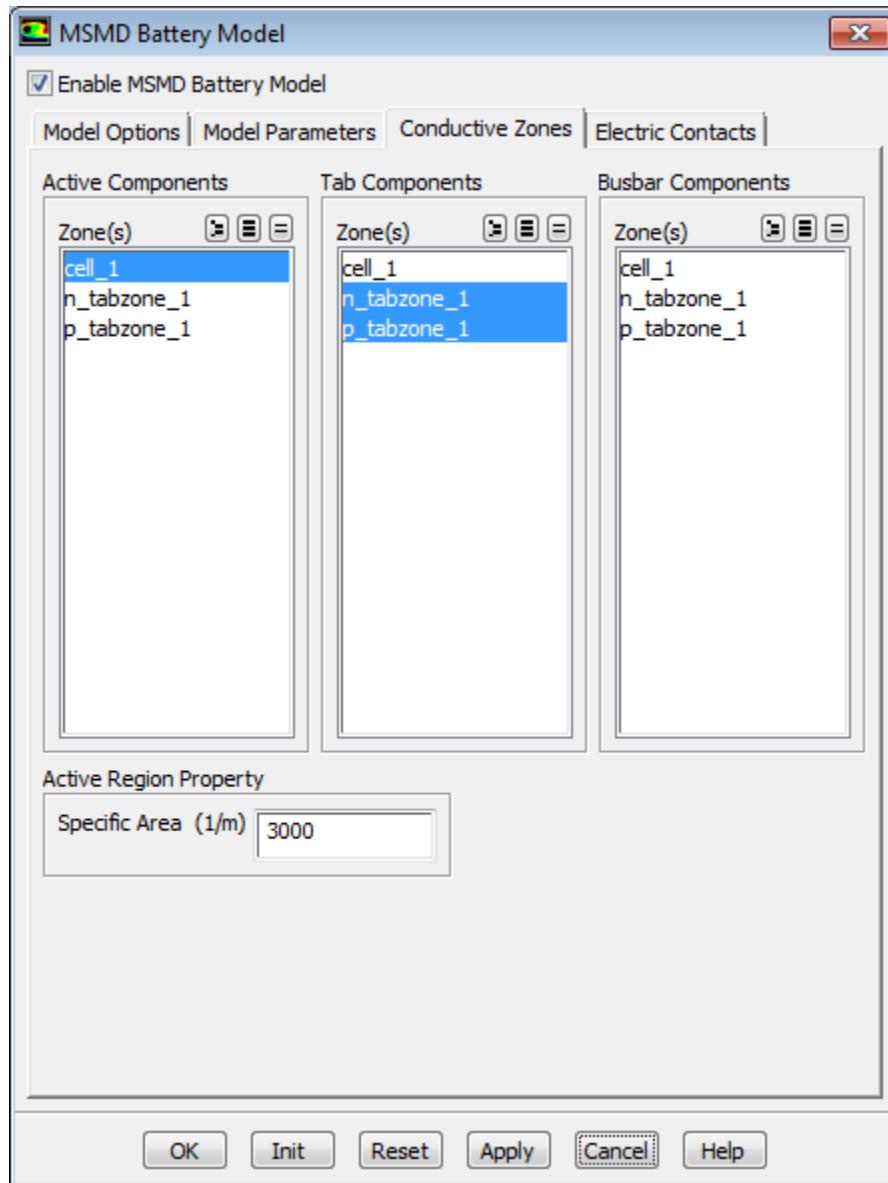
### Reference Capacity

is the battery capacity ( $\text{Ah}/\text{m}^2$ ).

### 3.2.4.3. Specifying Conductive Zones

The **Conductive Zones** tab of the **MSMD Battery Model** dialog box allows you to specify the active, tab, and busbar zones.

**Figure 3.10: The MSMD Battery Model Dialog Box (Conductive Zones Tab)**



In the **Conductive Zones** tab, select the appropriate zones for the **Active Components**, **Tab Components**, and, if you simulate the battery pack with the real battery connections, the **Basbar Components**. If you simulate the battery pack with the virtual battery connections, do not select any zones under **Basbar Components**.

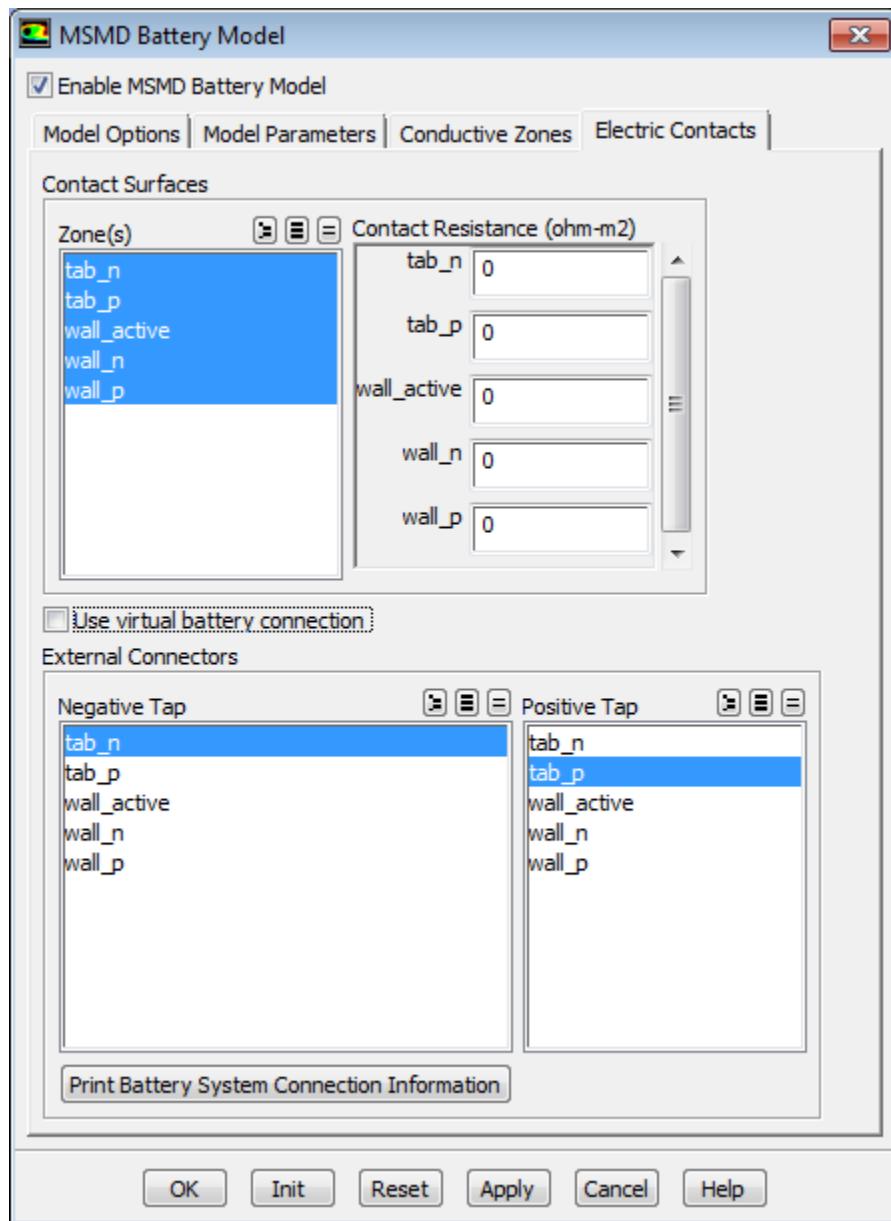
Battery cells are modeled as active zones; and battery tabs and busbars are modeled as passive zones. Electrochemical reactions occur only in the active zones. The potential field is solved in both active and passive zones.

If you are using the **NTGK Empirical Model**, you can also specify the specific area of the electrode sandwich sheet in the battery using the **Specific Area** control under **Active Region Property**.

### 3.2.4.4. Specifying Electric Contacts

The **Electric Contacts** tab of the **MSMD Battery Model** dialog box allows you to set the properties of the **Contact Surfaces** and the **External Connectors**.

**Figure 3.11: The MSMD Battery Model Dialog Box (Electric Contacts Tab)**



If you want to consider contact resistance, select zones for the **Contact Surfaces** and set the **Contact Resistance** for any selected contact surface.

When using real connections, define the system negative and positive tabs under **External Connectors**.

When using the virtual connection option in a battery pack simulation, select the **Use virtual battery connection** check box and in the **Specify connection file** text-entry box that appears, enter the filename containing the virtual connection definition.

The format for the connection definition file is as follows:

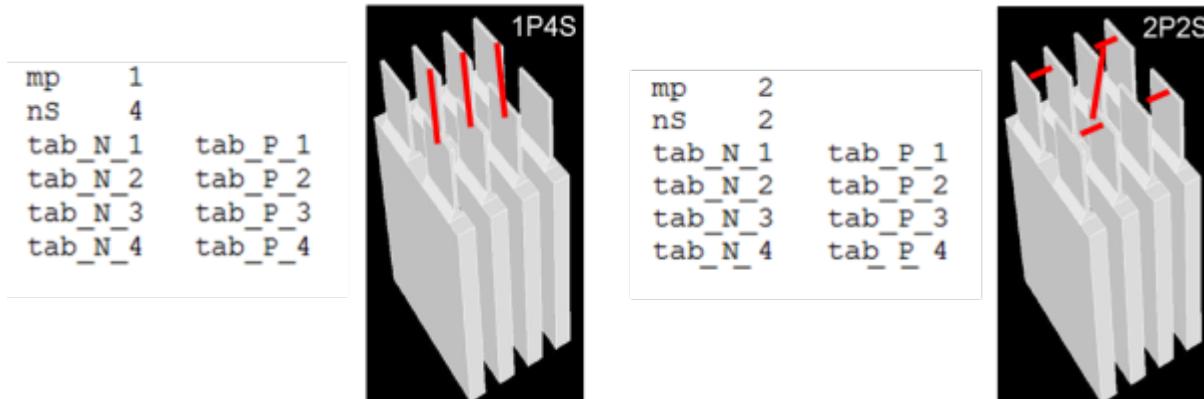
```
mp 1
nS 4
tab_N_1 tab_P_1
tab_N_2 tab_P_2
tab_N_3 tab_P_3
tab_N_4 tab_P_4
```

where:

- The first line specifies the number of parallel batteries in a stage (mP).
- The second line specifies the number of stages in series in a pack (nS).
- The remaining lines contain pairs of the face names of the negative and positive tabs for each battery. The battery data are listed in the following order: 1P1S ... 1PnS to mP1S ... mPnS.

For the above battery connection file example, the battery solver establishes the 1P4S connection pattern.

The following figure shows the different resulting connection patterns for the different connection data.

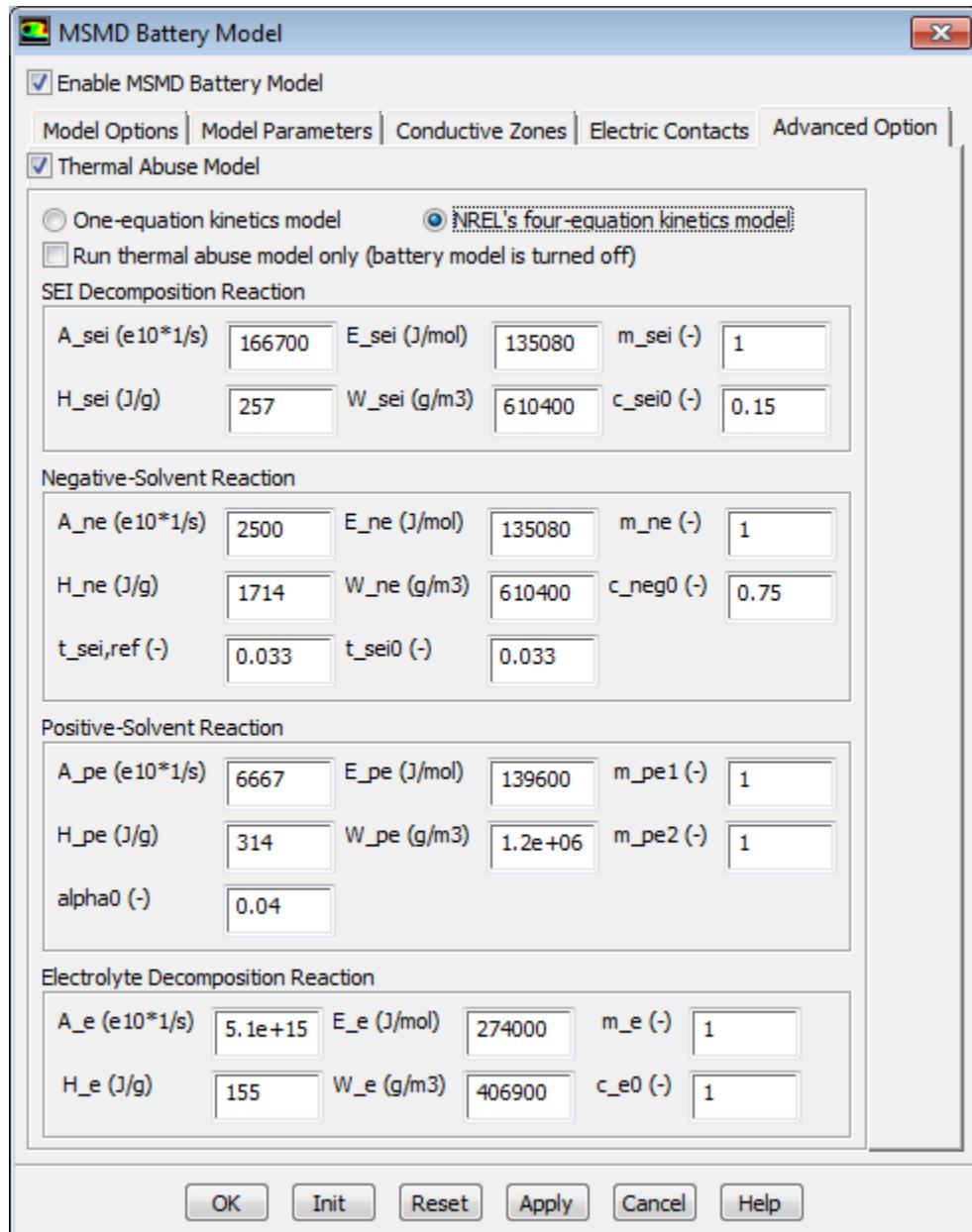


Once all conductive zones and the positive and negative tabs have been defined, ANSYS Fluent automatically detects the battery network connection.

After you have defined all the conductive zones or modified zone related data, click **Print Battery System Connection Information**, and review the battery connection information printed in the ANSYS Fluent console window for any errors. If necessary, re-define the connections in the **Conductive Zones** and **Electric Contacts** tabs.

### 3.2.4.5. Specifying Advanced Option

The **Advanced Option** tab of **MSMD Battery Model** dialog box allows you to enable **Thermal Abuse Model**, select the thermal abuse kinetics model, and define the corresponding model parameters.

**Figure 3.12: The MSMD Battery Model Dialog Box (Advanced Option Tab)**

When you select **One-equation kinetics model**, you can specify the battery constants  $A$ ,  $E$ ,  $H$ ,  $W$ ,  $m$ , and  $n$  used in [Equation 3.24 \(p. 95\)](#) through [Equation 3.25 \(p. 95\)](#), as well as the initial value for  $\alpha$ .

When you select **Four-equation kinetics model**, you can specify the battery constants  $A$ ,  $E$ ,  $H$ ,  $W$ , and  $m$  used in [Equation 3.26 \(p. 96\)](#) through [Equation 3.30 \(p. 97\)](#), as well as the initial values for  $c_{sei}$ ,  $c_{ne}$ ,  $\alpha$ , and  $c_e$ .

For both models, you can select **Run thermal abuse model only (battery model is turned off)** if you want to solve the energy equation ([Equation 3.1 \(p. 85\)](#)) alone. Otherwise, the two battery potential equations ([Equation 3.2 \(p. 85\)](#)) will also be solved.

---

### Note

- The kinetics parameters in thermal abuse models are specific to the battery material. The default settings may not be appropriate for your battery. You must provide parameter data for your own battery.
  - For the One-Equation thermal abuse model, you can use the ANSYS Fluent parameter estimation tool available through the TUI to compute the kinetics parameters from the battery oven testing data. The entry fields in the **MSMD Battery Model** dialog box will be automatically populated with fitted results for the kinetics parameters. For details, see [Using the Dual-Potential MSMD Battery Model Text User Interface \(p. 124\)](#).
- 

For more information, see [Thermal Abuse Model \(p. 95\)](#).

### 3.2.4.6. Specifying External and Internal Short-Circuit Resistances

#### External Short-Circuit

To define the external short-circuit: In the **MSMD Battery Model** dialog box, under the **Model Options** tab, select **Specified Resistance** and specify a value for **External Resistance**. See [Specifying Battery Model Options \(p. 100\)](#) for details.

#### Internal Short-Circuit

The intensity of the internal short-circuit is determined by the value of volumetric contact resistance  $r_c/a$  ([External and Internal Electric Short-Circuit Treatment \(p. 94\)](#)). In the ANSYS Fluent battery model, this value is saved in a predefined user-defined memory allocation, Short Circuit Resistance. You can modify the value of this user-defined memory allocation to simulate the internal short circuit.

By default, the contact resistance is set to a very large number, so that  $j_{short}=0$  and  $\dot{q}_{short}=0$ , that is, there is no internal short.

You can simulate an internal short-circuit using either of the following two ways:

- **Patching:** Patch the contact resistance to the short circuit zone. The short circuit zone can be defined using the marking feature available in the **Region Adaption** dialog box. See [Performing Region Adaption in the Fluent User's Guide](#) for more details.
- **Using UDF:** Use `DEFINE_ADJUST` or `DEFINE_ON_DEMAND` UDFs to specify the contact resistance using the UDM macro, `C_UDMI(c, t, SHORT_R)`. The contact resistance could be a function of location and time. See the [Fluent Customization Manual](#) for details.

### 3.2.5. Initializing the Battery Model

There are two ways to initialize the battery model:

- Initialize the flow field from the **Solution Initialization** task page. The battery model will be initialized also.
- Initialize the battery model in isolation by clicking the **Init** button in the **MSMD Battery Model** dialog box. Other flow variables remain untouched.

Use the second method if you want to freeze the flow field and use a different battery model to simulate the battery field. When switching from one battery submodel to another, the battery model must be re-initialized. The **Init** button in the **MSMD Battery Model** dialog box provides a convenient way to accomplish this if you do not want to initialize the flow field.

### 3.2.6. Modifying Material Properties

When using the MSDS battery models in ANSYS Fluent, you must define the UDS diffusivity of the cell, tab, and busbar materials.

#### 3.2.6.1. Specifying UDS Diffusivity for the Active Material

The UDS diffusivity of the active material must be defined using the **defined-per-uds** method as follows:

1. In the **Create/Edit Materials** dialog box, select **defined-per-uds** from the drop-down list for **UDS Diffusivity**.

The **UDS Diffusion Coefficients** dialog box is opened listing the **uds-0** and **uds-1** scalar diffusion coefficients defined in ANSYS Fluent. Both scalars are set at the default values of  $1e+07$ .

2. Modify the default values for **uds-0** and **uds-1** (if necessary).
3. For anisotropic material, select the **orthotropic**, **cyl-orthotropic**, or **anisotropic** option from the **Coefficient** drop-down list and specify the diffusion coefficients as described in [Fluent Customization Manual](#).

Note, that if the UDS Diffusion Coefficients are defined through the **defined-per-uds** option, the Fluent solver does not use the value for **Electrical Conductivity**.

#### 3.2.6.2. Specifying UDS Diffusivity for the Passive Material

The UDS diffusivity of the passive material must be defined using user-defined functions (UDF) as follows:

1. In the **Create/Edit Materials** dialog box, in the **Properties** group box, select **user-defined** from the drop-down list for **UDS Diffusivity**.
2. In the **User-Defined Functions** dialog box, hook the `battery_e_cond :: lib_caebat` predefined function provided by ANSYS Fluent. Note that this function cannot be modified.
3. In the **Create/Edit Materials** dialog box, in the **Properties** group box, specify the value for **Electrical Conductivity** (Fluent solver uses this value for the definition of `battery_e_cond`).

#### 3.2.6.3. Defining Different Materials for Positive and Negative Electrodes

If electric conductivities  $\sigma_+$  and  $\sigma_-$  are different for the positive and negative electrodes, you must define two different active materials, one for the odd zones and another for the even zones as follows:

1. When defining a material for the odd zones, in the **UDS Diffusion Coefficients** dialog box, enter the value of  $\sigma_+$  for **uds-0** and the value of  $\sigma_-$  for **uds-1**.

2. When defining a material for the even zones, copy the odd-zone material that you have already defined and modify the UDS0 and UDS1 scalars by interchanging their values, that is entering the value of  $\sigma_-$  for **uds-0** and the value of  $\sigma_+$  for **uds-1**.
3. In the **Cell Zone Conditions** task page, attach the odd zone material to the odd zones, and even zone material to the even zones. To determine whether a zone belongs to the odd or even zone, use the **Print Battery System Connection Information** button in the **Electric Contracts** tab of the **MSMD Battery Model** dialog box and review the report printed in the console window.

### 3.2.7. Solution Controls for the Dual-Potential MSMD Battery Model

When you use the battery models, two potential equations are solved in addition to other fluid dynamic equations, depending on the type of simulation. The two potential equations (**Potential Phi+** and **Potential Phi-**) are listed in the **Equations** dialog box, where they can be enabled/disabled during the solution process.

 **Solution** →  **Solution Controls** → **Equations...**

Also, keep in mind the **Advanced Solution Controls** dialog box.

 **Setup** →  **Solution Controls** → **Advanced...**

### 3.2.8. Postprocessing the Dual-Potential MSMD Battery Model

You can perform postprocessing using standard ANSYS Fluent quantities and by using user-defined scalars and user-defined memory allocations. When using the Battery model, the following additional variables will be available for postprocessing:

- Under **User-Defined Scalars...**
  - **Potential Phi+**
  - **Potential Phi-**
  - **Diff Coef of Potential Phi+**
  - **Diff Coef of Potential Phi-**
- Under **User-Defined Memory...**
  - **Transfer Current Density**
  - **X Current Density**
  - **Y Current Density**
  - **Z Current Density**
  - **Magnitude of Current Density**
  - **Volumetric Ohmic Source** (the energy source due to the electric Joule heating)
  - **Total Heat Generation Source**

- **Cell Voltage**
- **Activation Over-Potential** (the net electrode potential change across the anode and cathode of the system when there is a current flowing through the system, that is,  $\phi_c - \phi_a - U$  (Volts))
- **Depth of Discharge**
- **U Function** (the NTGK model only)
- **Y Function** (the NTGK model only)
- **Volumetric Current Source** (the transfer current density  $j$  in [Equation 3.2 \(p. 85\)](#).)
- **X Current Density for phi+**
- **Y Current Density for phi+**
- **Z Current Density for phi+**
- **X Current Density for phi-**
- **Y Current Density for phi-**
- **Z Current Density for phi-**
- **Short Circuit Resistance** (internal short circuit intensity)
- **Volumetric Short Current Source** (internal short current density)
- **Volumetric ECHEM Current Source** (Total exchange current density)
- **Short-Circuit Heat Source** (Joule heat due to internal short circuit)
- **Other**

---

#### Note

For reviewing simulation results in CFD Post for cases involving UDM or UDS, export the solution data as CFD-Post-compatible file (.cdat) in Fluent and then load this file into CFD-Post. This will ensure that the full variable set is available for post processing in CFD-Post.

---

### 3.2.9. User-Accessible Functions

You can incorporate your own correlations and data for the properties of the battery using the `cae_user.c` source code file.

The following listing represents a description of the contents of the `cae_user.c` source code file:

- `real CONDUCTIVITY_CELL(cell_t c, Thread *t, int i):` Returns values for the electrical conductivity for cells inside passive conductive zones, overwriting the values set in the **Create/Edit Materials** dialog box.
- `real CONTACT_RESIST_FACE(cell_t f, Thread *t):` Returns values for the electrical contact resistance, overwriting the values set in the **Electric Field** tab of the **MSMD Battery Model** dialog box.

- `real Compute_U (face_t f, Thread *t, real DOD, real dUDJ, real a[], real *dUDJ)`: Returns values for both U and the derivative of U with respect to J (dUDJ). Note that DOD represents the depth of discharge and a[] is the set of U coefficients.
- `real Compute_Y (face_t f, Thread *t, real DOD, real b[])`: Returns values for Y. Note that DOD represents the depth of discharge and b[] is the set of Y coefficients.
- `void Compute_ECM_VOC_and_CR(real soc, real T, int mode, real *Voc, real *RRs, real *RR1, real *RR2, real *CC1, real *CC2)`: Returns values for ECM model-related parameters. You can use this function to implement ECM parameters in any function form.
- `real Compute_OCP_NE(real x)`: Returns a value for open circuit potential at negative electrode ( $\varphi_-$  in [Equation 3.2 \(p. 85\)](#)). By default, it is computed using functions in [\[4\] \(p. 131\)](#).
- `real Compute_OCP_PE(real x)`: Returns a value for open circuit potential at positive electrode ( $\varphi_+$  in [Equation 3.2 \(p. 85\)](#)). By default, it is computed using functions in [\[4\] \(p. 131\)](#).
- `real Compute_kappa(real ce, real T)`: Returns a value for  $k$  ([Equation 3.19 \(p. 90\)](#)). By default, it is computed using functions in [\[4\] \(p. 131\)](#).
- `real Compute_Diff0_ce(real ce, real T)`: Returns a value for  $D^{eff}$  ([Equation 3.19 \(p. 90\)](#)). By default, it is computed using functions in [\[4\] \(p. 131\)](#).
- `real Compute_tplus(real ce, real T)`: Returns a value for  $t_+^0$  ([Equation 3.19 \(p. 90\)](#)). By default, it is computed using functions in [\[4\] \(p. 131\)](#).
- `real Compute_dlnfdlnce(real ce, real T)`: Returns a value for the term  $\frac{d\ln f}{d\ln c_e}^\pm$  from [Equation 3.19 \(p. 90\)](#). By default, this term is not considered.
- `real Compute_entropy_heat(real soc)`: Returns the entropic heat. By default, it is not considered in all MSMD submodels. You can include it by defining  $\frac{dU}{dT}$  in this UDF.
- `void user_defined_echem_model(int zero_start, int mode, real temperature, real voltage, real current, real dttime, real *j_tmp, real *Qe_tmp, real *voltage_end)`: Defines the user-defined electrochemical model. A template for the user implementation of the NTGK model is provided as an example.

For more information, see the following sections:

### [3.2.9.1. Compiling the Customized Battery Source Code](#)

## **3.2.9.1. Compiling the Customized Battery Source Code**

This section includes instructions on how to compile a customized Battery user-defined module. Note that you can also refer to the file `INSTRUCTIONS-CLIENT` that comes with your distribution (see `addons/msmdbatt`).

---

### **Important**

It is assumed that you have a basic familiarity with compiling user-defined functions (UDFs). For an introduction on how to compile UDFs, refer to the separate [Fluent Customization Manual](#).

---

You will first want to use a local copy of the `msmdbatt` directory in the `addons` directory before you recompile the Battery module.

### 3.2.9.1.1. Compiling the Customized Source Code Under Linux

1. Make a local copy of the `msmdbatt` directory. Do not create a symbolic link.

---

#### **Important**

The custom version of the library must be named according to the convention used by ANSYS Fluent: for example, `msmdbatt`.

---

2. Change directories to the `msmdbatt/src` directory.
3. Make changes to the `cae_user.c` file.
4. Edit the `makefile` located in the `src/` directory and make sure that the `FLUENT_INC` variable correctly refers to the current ANSYS Fluent installation directory. Be careful not to leave any trailing spaces when you make your changes.
5. Define the `FLUENT_ADDONS` environment variable to correspond to your customized version of the Battery module.
6. Change directories to the `msmdbatt/` directory.
7. Issue the following `make` command:

```
make FLUENT_INC=[ansys_inc/v170/fluent] FLUENT_ARCH=[arch]
-f Makefile-client
```

where `your_arch` is `lnamd64` on LINUX.

The following example demonstrates the steps required to set up and run a customized version of the Battery module that is located in a folder call `home/sample`:

- Make a directory (for example, `mkdir -p /home/sample`).
- Copy the default addon library to this location.

```
cp -RH [ansys_inc/v170/fluent]/fluent17.0.0/addons/msmdbatt
/home/sample/msmdbatt
```

- Using a text editor, make the appropriate changes to the `cae_user.c` file located in `/home/sample/msmdbatt/src/cae_user.c`
- Edit the `makefile` located in the `src/` directory and make sure that the `FLUENT_INC` variable correctly refers to the current ANSYS Fluent installation directory. Be careful not to leave any trailing spaces when you make your changes.
- Build the library.

```
cd /home/sample/msmdbatt
make FLUENT_INC=[ansys_inc/v170/fluent] FLUENT_ARCH=[arch]
-f Makefile-client
```

- Set the FLUENT\_ADDONS environment variable (using CSH, other shells will differ).

```
setenv FLUENT_ADDONS /home/sample
```

- Start ANSYS Fluent and load the customized module using the text interface command.

### 3.2.9.1.2. Compiling the Customized Source Code Under Windows

1. Open **Visual Studio.NET** at the DOS prompt.
2. Make sure that the FLUENT\_INC environment variable is correctly set to the current ANSYS Fluent installation directory (for example, ANSYS Inc\v170\fluent).
3. Make a local copy of the msmdbatt folder. Do not create a shortcut.
4. Enter the msmdbatt\src folder.
5. Make changes to the cae\_user.c file.
6. Define the FLUENT\_ADDONS environment variable to correspond to your customized version of the Battery module.
7. Return to the msmdbatt folder.
8. Issue the following command in the command window:

```
nmake /f makefile_master-client.nt
```

### 3.2.10. Using the Dual-Potential MSMD Battery Model Text User Interface

All of the features for the Battery Model that are available through the graphical user interface are also available through text user interface (TUI) commands. The TUI allows text commands to be typed directly in the ANSYS Fluent console window where additional information can be extracted and processed for more advanced analysis.

Once the battery module is loaded, you can access the text user interface through the Console Window by entering define models battery-model. A listing of the various text commands is as follows:

**battery-model/**

Enter the battery model menu.

**enable-battery-model?**

Enable/disable battery model.

**electric-field-model/**

Enter the electric field setup menu.

**conductive-regions**

Specify active conductive regions.

**contact-resistance-regions**

Specify contact resistance regions.

**current-tap**

Set cathode tap.

**voltage-tap**

Set anode tap.

**model-parameters**

Set battery model options (see [Specifying Battery Model Options \(p. 100\)](#) for definitions of the MSMD model general parameters).

**ntgk-parameters**

Set NTGK model-specific parameters (see [Inputs for the NTGK Empirical Model \(p. 107\)](#) for definitions of the MTGK model parameters).

**ecm-parameters**

Set ECM model-specific parameters (see [Inputs for the Equivalent Circuit Model \(p. 108\)](#) for definitions of the ECM model parameters).

**newman-parameters**

Set Newman model-specific parameters (see [Inputs for the Equivalent Circuit Model \(p. 108\)](#) for definitions of the Newman model parameters).

**parameter-estimation-tool**

Compute the model parameters from battery's testing data. You can use the estimation tool to compute model parameters for the NTGK or ECM model, or for the One-Equation thermal abuse model. You must provide battery's testing data as a text file in a specific format for each model. When you enable this tool and select the model option, Fluent will show the required format for the input file in the console. You can contact ANSYS Technical Support for clarifications on how to use this tool.



---

---

## **Chapter 4:Tutorial: Simulating a Single Battery Cell Using the MSMD Battery Model**

---

A tutorial is available that provides an example of how to set up a battery cell simulation using the three submodels that are available in ANSYS Fluent. The latest update of this tutorial is available on the ANSYS Customer Portal. To access tutorials and their input files on the ANSYS Customer Portal, go to <http://support.ansys.com/training>.



---

---

## **Chapter 5:Tutorial:Simulating a 1P3S Battery Pack Using the MSMD Battery Model**

---

A tutorial is available that provides an example of how to set up a battery pack (battery system connected in parallel / series pattern) simulation in ANSYS Fluent. The latest update of this tutorial is available on the ANSYS Customer Portal. To access tutorials and their input files on the ANSYS Customer Portal, go to <http://support.ansys.com/training>.



---

# Bibliography

- [1] H. Gu. "Mathematical Analysis of a Zn/NiOOH Cell". *J. Electrochemical Soc.*. Princeton, NJ. 1459-1. 464. July 1983.
- [2] U. S. Kim, C. B. Shin, and C.-S. Kim. "Effect of Electrode Configuration on the Thermal Behavior of a Lithium-Polymer Battery". *Journal of Power Sources*. Princeton, NJ. 180. 909–916. 2008.
- [3] Tiedemann and Newman. S.Gross Editor. "Battery Design and Optimization". *Proceedings, Electrochemical Soc.* Princeton, NJ. 79-1. 39. 1979.
- [4] L. Cai and R.E. White. "Reduction of Model Order Based on Proper Orthogonal Decomposition for Lithium-Ion Battery Simulations". *J. of Electrochemical Soc.* 156(3). A154-A161. 2009.
- [5] M. Chen and G. A. Rincon-Mora. "Accurate Electrical Battery Model Capable of Predicting Runtime and I-V Performance". *IEEE Trans. On Energy Conversion*. Vol. 21. No.2. A154-A161. June 2006.
- [6] M. Doyle, T.F. Fuller and J. Newman. "Modeling of Galvanostatic Charge and Discharge of the Lithium/Polymer/Insertion Cell". *J. of Electrochemical Soc.*. Vol 140, No. 6. 1526-1533. 1993.
- [7] G-H. Kim, A. Peraran, R. Spotnitz. "A Three-dimensional thermal abuse model for lithium-ion cells". *Journal of Power Sources*. 170 (2). 476-489. 2007.
- [8] G-H. Kim et al. "Multi-Domain Modeling of Lithium-Ion Batteries Encompassing Multi-Physics in Varied Length Scales". *J. of Electrochemical Soc.*. 158 (8). A955-A969. 2011.
- [9] U. S. Kim et al. "Modeling the Dependence of the Discharge Behavior of a Lithium-Ion Battery on the Environmental Temperature". *J. of Electrochemical Soc.*. 158 (5). A611-A618. 2011.
- [10] U. S. Kim et al. "Effect of electrode configuration on the thermal behavior of a lithium-polymer battery". *Journal of Power Sources*. 180 (2). 909-916. 2008.
- [11] K. H. Kwon et al. "A Two-dimensional Modeling of a Lithium-polymer battery". *Journal of Power Sources*. 163. 151-157. 2006.
- [12] K. Smith and C.Y. Wang. "Solid-state Diffusion Limitations on Pulse Operation of a Lithium ion Cell for Hybrid Electric Vehicles". *Journal of Power Sources*. 161. 628-639. 2006.



---

---

## **Part III: ANSYS Fluent Continuous Fiber Module**

[Using This Manual \(p. cxxxv\)](#)

[1. Introduction \(p. 137\)](#)

[2. Continuous Fiber Model Theory \(p. 139\)](#)

[3. Using the Continuous Fiber Module \(p. 155\)](#)

[Bibliography \(p. 191\)](#)

---

---

---

# Using This Manual

---

This preface is divided into the following sections:

**1. The Contents of This Manual**

## 1. The Contents of This Manual

The ANSYS Fluent Continuous Fiber Model Manual tells you what you need to know to model melt and dry spinning processes with ANSYS Fluent. In this manual, you will find background information pertaining to the model, a theoretical discussion of the model used in ANSYS Fluent, and a description of using the model for your CFD simulations.

This part is divided into the following chapters:

- [Introduction \(p. 137\)](#)
- [Continuous Fiber Model Theory \(p. 139\)](#)
- [Using the Continuous Fiber Module \(p. 155\)](#)
- [Bibliography \(p. 191\)](#)



---

## Chapter 1: Introduction

---

The continuous fiber model is provided as an add-on module with the standard ANSYS Fluent licensed software.

Several fiber spinning techniques exist in industrial fiber production. The most common types are melt spinning and dry spinning.

In melt spinning, the polymer is heated above its melting point and extruded in a liquid state through nozzles into a vertical spinning chamber. The molten polymer is processed in an inert gas environment, such as nitrogen, then extruded at high pressure and a constant rate into a cooler air stream, thus solidifying the fiber filaments.

In dry spinning, the liquefaction of the polymer is obtained by dissolving it in a suitable solvent. This technique often is applied to polymers that are destroyed thermally before reaching its melting point or if the production process leads to a solvent/polymer mixture. In the spinning chamber, the solvent vaporizes by drying with a hot air stream. The solidification ensures that the fiber is nearly free of solvent.

ANSYS Fluent's continuous fiber model allows you to analyze the behavior of fiber flow, fiber properties, and coupling between fibers and the surrounding fluid due to the strong interaction that exists between the fibers and the surrounding gas.

This document describes the ANSYS Fluent Continuous Fiber model. [Continuous Fiber Model Theory \(p. 139\)](#) provides theoretical background information. Instructions for getting started with the model are provided in [Using the Continuous Fiber Module \(p. 155\)](#).



---

## Chapter 2: Continuous Fiber Model Theory

---

This chapter presents an overview of the theory and the governing equations for the mathematical model and ANSYS Fluent's capabilities to predict melt and dry spinning processes.

- 2.1. Introduction
- 2.2. Governing Equations of Fiber Flow
- 2.3. Discretization of the Fiber Equations
- 2.4. Numerical Solution Algorithm of Fiber Equations
- 2.5. Residuals of Fiber Equations
- 2.6. Coupling Between Fibers and the Surrounding Fluid
- 2.7. Fiber Grid Generation
- 2.8. Correlations for Momentum, Heat and Mass Transfer
- 2.9. Fiber Properties
- 2.10. Solution Strategies

### 2.1. Introduction

ANSYS Fluent's Continuous Fiber Model uses a one-dimensional approach used to predict the flow in fibers and to predict the flow field in the spinning chamber.

In melt spinning, where the extruded molten polymer is sent through the nozzles into the spinning chamber, the velocity of the liquid jet increases due to gravity and the tensile force, which is applied at the take-up point of the fibers. The conservation of mass leads to a decrease in the cross-section of the jet up to the final diameter. The molten polymer is cooled by an air stream until the solidification temperature is reached.

In dry and melt spinning, production of hundreds or thousands of fibers in a spinning chamber leads to strong interaction between the fibers and the surrounding gas, requiring a coupled calculation procedure for the fibers and the fluid flow in the spinning chamber.

### 2.2. Governing Equations of Fiber Flow

Mass conservation of a fiber element is written as

$$\frac{d}{dz} \left( \rho_f \vec{u}_f \vec{A}_f Y_s \right) = -\pi d_f \dot{m}_s'' \quad (2.1)$$

where

$\rho_f$  = fiber density

$\vec{u}_f$  = fiber velocity vector

$\vec{A}_f$  = surface area vector of the fiber surface parallel to the flow direction

$Y_s$  = mass fraction of the solvent  $s$  in the fiber

$\dot{m}_s''$  = evaporated mass flow rate of the solvent  $s$

$d_f$  = fiber diameter

$z$  = coordinate along the fiber

$\dot{m}_s''$  is calculated using a film theory.

$$\dot{m}_s'' = M_s c \beta \ln \left( \frac{1 - \psi_{s,g}}{1 - \psi_{s,I}} \right) \quad (2.2)$$

where

$\beta$  = mass transfer coefficient estimated from an appropriate correlation (see [Correlations for Momentum, Heat and Mass Transfer \(p. 147\)](#))

$M_s$  = solvent's molecular weight

$c$  = molar concentration of the surrounding gas

$\psi_{s,g}$  = mole fraction of the solvent vapor in the surrounding gas

At the fiber surface, the mole fraction of the solvent in the gas  $\psi_{s,I}$  is related to the solvent mass fraction in the fiber  $Y_s$  by the vapor-liquid equilibrium equation given by Flory [1] (p. 191),

$$\psi_{s,I} = \frac{p_{s,vap}}{p} Y_s e^{1-Y_s + \chi(1-Y_s)^2} \quad (2.3)$$

where

$\chi$  = Flory-Huggins parameter

$p$  = absolute pressure in the surrounding flow

$p_{s,vap}$  = saturation vapor pressure of the solvent

These equations are used only when dry spun fibers have been selected.

The formation of fibers is based on tensile forces in the fiber that are applied at the take-up point and result in the drawing and elongation of the fiber.

A force balance for a differential fiber element gives the equation of change of momentum in the fiber.

$$\frac{d(\rho_f u_f u_f \vec{A}_f)}{dz} = \frac{d\vec{F}}{dz} + \vec{F}_{friction} + \vec{F}_{gravitation} \quad (2.4)$$

The tensile force in the fiber changes due to acceleration of the fiber, friction force with the surrounding gas, and the gravitational forces.

The friction force with the surrounding gas is computed by

$$\vec{F}_{friction} = \frac{1}{2} \rho c_{f,ax} \pi d_f \left| \vec{u}_f - \vec{u}_{par} \right| \left( \vec{u}_f - \vec{u}_{par} \right) \quad (2.5)$$

where

$\rho$  = gas density

$c_{f,ax}$  = axial friction factor parallel to the fiber

$\vec{u}_{par}$  = gas velocity parallel to the fiber

The gravitational force is computed from

$$\vec{F}_{gravitation} = \rho_f \frac{\pi}{4} d_f^2 \vec{g} \cdot \vec{n}_f \quad (2.6)$$

where  $\vec{n}_f$  is the direction vector of the fiber element.

The tensile force  $\vec{F}$  is related to the components of the stress tensor by

$$\vec{F} = \vec{A}_f (\tau_{zz} - \tau_{rr}) \quad (2.7)$$

Neglecting visco-elastic effects and assuming Newtonian flow one can obtain

$$\tau_{zz} = 2\eta_f \frac{du_f}{dz} \quad (2.8)$$

$$\tau_{rr} = -\eta_f \frac{du_f}{dz} \quad (2.9)$$

leading to

$$\vec{F} = 3\vec{A}_f \eta_f \frac{du_f}{dz} \quad (2.10)$$

The elongational viscosity is estimated by multiplying the zero shear viscosity  $\eta_f$  by three.

The transport of enthalpy in and to a differential fiber element is balanced to calculate the fiber temperature along the spinning line.

$$\begin{aligned} \frac{d}{dz} \left( \rho_f \vec{u}_f \cdot \vec{A}_f h_f \right) &= \frac{d}{dz} \left( \lambda_f \vec{A}_f \frac{dT_f}{dz} \right) + \pi d_f \left( \alpha (T - T_f) - \dot{m}_s'' h_{s,v} \right) \\ &+ \dot{Q}_{viscous\ heating} + \dot{Q}_{radiation,abs} - \dot{Q}_{radiation,emission} \end{aligned} \quad (2.11)$$

where

$h_f$  = fiber enthalpy

$\lambda_f$  = fiber thermal conductivity

$T_f$  = fiber temperature

$h_{s,v}$  = enthalpy of the solvent vapor

$\alpha$  = heat transfer coefficient

In the case of a melt spinning process,  $\dot{m}_s''$  is zero because there is no mass transfer. The term for heat generation due to viscous heating is derived from the fluid mechanics of cylindrical flow to be

$$\dot{Q}_{viscous\ heating} = \frac{\pi}{4} d_f^2 \left( 4 \left( \frac{du_f}{dz} \right)^2 - \frac{2}{3} \dot{m}_s'''^2 \right) \quad (2.12)$$

Radiation heat exchange is considered by the last two terms

$$\dot{Q}_{radiation,abs} = d_f \varepsilon_f G \quad (2.13)$$

$$\dot{Q}_{radiation,emission} = \pi d_f \varepsilon_f \sigma T_f^4 \quad (2.14)$$

where

$G$  = thermal irradiation

$\varepsilon_f$  = fiber's emissivity

$\sigma$  = Boltzman constant

The fiber enthalpy  $h_f$  is related to the fiber temperature  $T_f$  as follows

$$h_f = \int_{T_{ref}}^{T_f} ((1-Y_s)C_{p_p} + Y_s C_{p_s}) dT \quad (2.15)$$

It uses  $C_{p_p}$  the specific heat capacity of the polymer and  $C_{p_s}$  the specific heat capacity of the solvent in the fiber.

The enthalpy of the solvent vapor at a given temperature  $T_v$  depends on the heat of vaporization  $\Delta h_s$ , given at the vaporization temperature  $T_{vap}$ , and is computed from

$$h_{s,v} = \int_{T_{ref}}^{T_{vap}} C_{p_{s,l}} dT + \Delta h_s|_{T_{vap}} + \int_{T_{vap}}^{T_v} C_{p_{s,v}} dT \quad (2.16)$$

where

$C_{p_{s,l}}$  = specific heat capacity of the solvent liquid

$C_{p_{s,v}}$  = specific heat capacity of the solvent vapor

## 2.3. Discretization of the Fiber Equations

The governing [Equation 2.1 \(p. 139\)](#), [Equation 2.4 \(p. 140\)](#), and [Equation 2.11 \(p. 141\)](#) have the form of convection diffusion equations and are discretized using a finite volume scheme of Patankar [\[5\] \(p. 191\)](#).

The convective terms in [Equation 2.1 \(p. 139\)](#), [Equation 2.4 \(p. 140\)](#), and [Equation 2.11 \(p. 141\)](#) are discretized using first order upwinding, central differencing or the DISC scheme [\[6\] \(p. 191\)](#). While the first two schemes are well described in the ANSYS Fluent [User's Guide](#), the reader is referred to [\[6\] \(p. 191\)](#) to learn more about the DISC scheme. It is the recommended scheme because it provides outstanding numerical stability combined with second order accuracy. The diffusion terms are discretized with second order accuracy. All other terms are treated as source terms and linearized according to Patankar [\[5\] \(p. 191\)](#).

For more information, see the following section:

### 2.3.1. Under-Relaxation

Because the fiber equations are nonlinear, the change of the solution variable  $\phi$  has to be controlled. This is achieved by under-relaxation. The new value of  $\phi$  in each cell depends to some degree upon the old value  $\phi_{old}$  and the change in  $\phi$ ,  $\Delta\phi$ . It is computed for a given under-relaxation factor  $\alpha$  as follows:

$$\phi = \phi_{old} + \alpha \Delta\phi \quad (2.17)$$

## 2.4. Numerical Solution Algorithm of Fiber Equations

Each governing differential equation is discretized into a set of algebraic equations that are solved using the tri-diagonal matrix algorithm. All differential equations for conservation of mass, momentum, energy, and (when appropriate) for solvent in the fiber are solved sequentially (that is, segregated from one another). Because the governing equations are coupled and nonlinear, several iterations have to be performed to obtain a converged solution. The solution process consists of several steps outlined below:

1. The fiber properties are updated based on the initialized or the current solution.
2. The friction factors for momentum exchange between the fibers and surrounding fluid are computed based on current values of fiber and fluid velocities.
3. The fiber momentum equation is solved and current values of the mass fluxes in the fiber are used.
4. The heat transfer coefficients are computed using Reynolds numbers from the beginning of the iteration loop.
5. The fiber energy equation is solved.
6. In the case of dry spun fibers, the equation for the mass fraction of the solvent is solved. First, the mass transfer coefficient is updated. The evaporated (condensed) mass is computed based on the vapor liquid equilibrium at the beginning of the iteration loop. Finally, the governing equation is solved.
7. The mass fluxes and the diameter of the fiber cells are updated.
8. A check for convergence of the equation set is made.

These steps are continued until the convergence criteria are met for all equations of the considered fiber or until the number of iterations exceed the given limit.

This solution algorithm is applied to all defined fibers.

## 2.5. Residuals of Fiber Equations

The solution algorithm for the fiber equations requires a means for checking convergence of the solution. In the fiber model a simple residual is used for this purpose.

The conservation equation for a general variable  $\phi$  at a cell  $P$  can be written as

$$a_P \phi_P = \sum_{nb} a_{nb} \phi_{nb} + S_c + S_p \phi_P \quad (2.18)$$

where

$a_P$  = center coefficient

$a_{nb}$  = influence coefficients for the neighboring cells

$S_c$  = constant part of the source term

$S_p$  = linear part of the source term

The residual  $R^\phi$  computed by the fiber model is the imbalance in Equation 2.18 (p. 143) summed over all fiber cells.

$$R^\phi = \sum_{fiber\ cells} \left| \sum_{nb} a_{nb} \phi_{nb} + S_c + S_p \phi_p - a_p \phi_p \right| \quad (2.19)$$

This is called the absolute residual. Relative residuals are defined as the change of the absolute residuals between two subsequent iterations divided by the absolute residual.

$$\hat{R}^\phi = \frac{R^\phi_{iterationN} - R^\phi_{iterationN-1}}{R^\phi_{iterationN}} \quad (2.20)$$

## 2.6. Coupling Between Fibers and the Surrounding Fluid

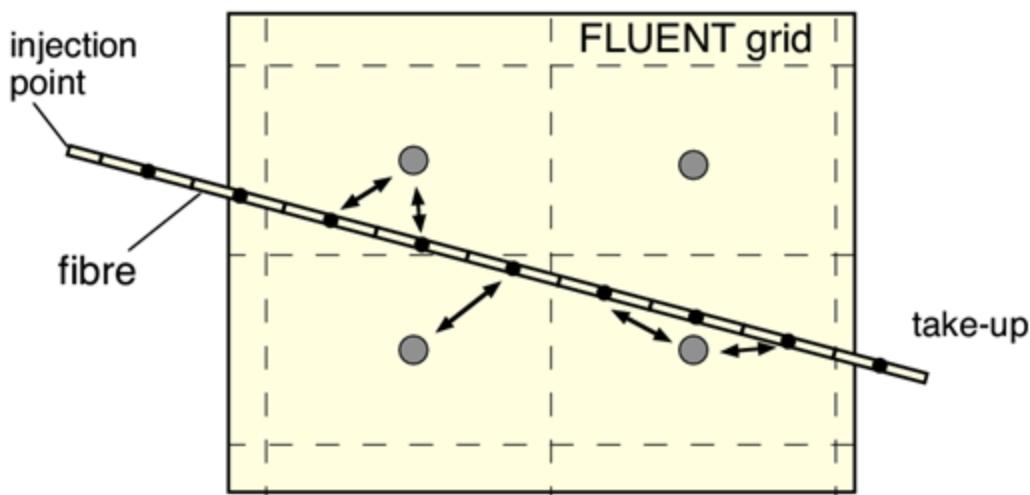
The fiber model accounts for all mass, momentum, and heat gained or lost by the fibers and can be incorporated into the subsequent calculations of the surrounding fluid performed by ANSYS Fluent. This means that you can incorporate the effects of the fibers on the surrounding fluid, while the surrounding fluid always impacts the fiber flow. This two-way coupling is achieved by alternatively solving the fiber and the surrounding fluid flow equations until both phases have converged. The interphase exchange of heat, mass, and momentum from the fibers to the surrounding fluid is depicted qualitatively in [Figure 2.1: Fiber Grid Penetrating Grid of the Gas Flow \(p. 144\)](#).

For more information, see the following sections:

- [2.6.1. Momentum Exchange](#)
- [2.6.2. Mass Exchange](#)
- [2.6.3. Heat Exchange](#)
- [2.6.4. Radiation Exchange](#)
- [2.6.5. Under-Relaxation of the Fiber Exchange Terms](#)

### 2.6.1. Momentum Exchange

**Figure 2.1: Fiber Grid Penetrating Grid of the Gas Flow**



Momentum transfer from the fibers to the surrounding fluid is computed in ANSYS Fluent by considering the change of momentum of the fiber as it crosses each control volume in the ANSYS Fluent model. It is computed as

$$\vec{F}_c = \sum_{fibers} \left( \frac{\rho}{2} \pi d_f l_{f,c} \left( c_{f,par} (\vec{u}_f - \vec{u}_{par})^2 - c_{f,lat} (\vec{u}_f - \vec{u}_{lat})^2 \right) + \vec{u}_f \pi d_f l_{f,c} \dot{m}_s'' \right) \quad (2.21)$$

where

$\rho$  = density of the fluid

$d_f$  = diameter of the fiber

$l_{f,c}$  = length of the fiber  $f$  in cell  $c$

$\vec{u}_f$  = velocity of the fiber

$\vec{u}_{par}$  = velocity of the fluid, parallel to the fiber

$\vec{u}_{lat}$  = velocity of the fluid, lateral to the fiber

$c_{f,par}$  = drag coefficient parallel to the fiber

$c_{f,lat}$  = drag coefficient lateral to the fiber

$\dot{m}_s''$  = evaporated mass flow rate of the solvent in the fiber

This momentum exchange appears as a momentum source in the surrounding fluid momentum balance and is taken into account during every continuous phase computation. It can be reported as described in [Exchange Terms of Fibers \(p. 185\)](#).

## 2.6.2. Mass Exchange

The mass transfer of evaporating solvent from the fibers in dry spinning applications is computed in ANSYS Fluent by balancing the evaporated mass of solvent in every fiber cell volume. It is computed as

$$M_c = \sum_{fibers} \pi d_f l_{f,c} \dot{m}_s'' \quad (2.22)$$

The mass exchange appears as a mass source in the continuity equation of the surrounding fluid as well as a source of chemical species of the solvent vapor defined by you. It is included in every subsequent calculation of the continuous phase flow field and is reported as described in [Exchange Terms of Fibers \(p. 185\)](#).

## 2.6.3. Heat Exchange

The heat transfer from the fibers to the surrounding flow is computed in ANSYS Fluent by balancing the change of the fiber energy as it crosses each control volume in the ANSYS Fluent model. It considers sensible as well as latent heat transfer due to evaporation of solvent in dry spinning applications and is computed as

$$Q_c = \sum_{fibers} \left( \pi d_f l_{f,c} \left( \alpha (T_f - T) + \dot{m}_s'' \left( \frac{1}{2} \vec{u}_f^2 + \int_{T_{ref}}^{T_f} C_{p,s,v} dT \right) \right) + \vec{u}_f \vec{F}_f \right) \quad (2.23)$$

where

$\alpha$  = heat transfer coefficient

$T_{ref}$  = reference temperature

$C_{p,s,v}$  = specific heat capacity of solvent vapor

$\vec{F}_f$  = momentum exchange of fiber  $f$

This heat exchange appears as a source in the surrounding fluid energy balance and is taken into account during every continuous phase computation. It can be reported as described in [Exchange Terms of Fibers \(p. 185\)](#).

## 2.6.4. Radiation Exchange

The fibers participate in radiation exchange by absorbing irradiation energy from the surrounding flow irradiation and by emitting irradiation at the fiber temperature. This effect on the irradiation of the surrounding flow is considered by computing the absorbed and emitted energy of the fibers in each cell as

$$G_{abs} = \sum_{fibers} \frac{\varepsilon_f}{4} A_f G \quad (2.24)$$

$$G_{emiss} = \sum_{fibers} \frac{\varepsilon_f}{4} \frac{A_f}{\pi} \sigma T_f^4 \quad (2.25)$$

where

$A_f$  = fiber surface area

$\sigma$  = Boltzman constant

$G$  = irradiation of the surrounding flow

$\varepsilon_f$  = emissivity of the fiber

### Important

The transfer of the fiber radiation energy to the surrounding flow is only considered when the single-band Discrete-Ordinate Model is used in the ANSYS Fluent flow model. While radiation effects on the fibers are taken into account when the P1 model is used, there is no two-way coupling for this model.

## 2.6.5. Under-Relaxation of the Fiber Exchange Terms

Note that the exchange of momentum, heat, and mass from the fibers is under-relaxed during the calculation, so that

$$\vec{F}_{c,new} = \vec{F}_{c,old} + \alpha (\vec{F}_{c,calculated} - \vec{F}_{c,old}) \quad (2.26)$$

$$M_{c,new} = M_{c,old} + \alpha (M_{c,calculated} - M_{c,old}) \quad (2.27)$$

$$Q_{c,new} = Q_{c,old} + \alpha (Q_{c,calculated} - Q_{c,old}) \quad (2.28)$$

where  $\alpha$  is the under-relaxation factor for fibers that you can set in the **Solution Controls** task page. The default value for  $\alpha$  is 0.5. This value may be reduced in order to improve the stability of coupled calculations. Note that the value of  $\alpha$  does not influence the predictions obtained in the final converged solution.

## 2.7. Fiber Grid Generation

The fibers penetrate the grid of the surrounding flow field arbitrarily. Both grids are treated distinctly from each other (see [Figure 2.1: Fiber Grid Penetrating Grid of the Gas Flow \(p. 144\)](#)).

The fiber grids are generated by defining fiber injections. The fibers are considered to be straight lines between injection points and a take-up point. Each fiber is divided into a number of volume cells.

For the grid generation, the following grid types are available:

#### **equidistant**

All cells of the fiber have the same length.

#### **one-sided**

The cells are graded near the injection point of the fiber and change their size according to a specified growth factor.

#### **two-sided**

In addition to the injection point the cells can also be graded at the take-up point by specifying a second growth factor at the end of the fiber.

#### **three-sided**

The second point where the fiber cells are graded at the end can be moved to a local refinement point laying between injection and take-up point. This generates fibers with a mesh graded at the injection point and at the local refinement point.

All grid types, except the **equidistant** grid type, require the specification of a growth factor  $R$ , which is the ratio of two subsequent fiber grid cells. It refines the mesh for values larger than 1 and coarsens the mesh for values smaller than 1.

If a finite fiber volume cell spans across several ANSYS Fluent grid cells, a weighted average is used to estimate the corresponding variables of the surrounding flow. This averaging procedure considers the intersection point of each fiber volume cell with the boundaries of the ANSYS Fluent grid cell.

When computing the source terms in an ANSYS Fluent grid cell, only the part of each fiber volume cell that is inside the ANSYS Fluent grid cell is taken into account. This provides a proper computation of ANSYS Fluent fiber interactions even in hanging node adapted grids.

#### **Important**

If the grid is adapted, the data structures that include information about neighbor cells are not updated automatically. For this you have to reinitialize all fibers to start a new search of the neighboring ANSYS Fluent grid cells.

## **2.8. Correlations for Momentum, Heat and Mass Transfer**

The fiber model makes use of correlations to compute transfer of momentum, heat, and mass to the fibers. The fibers are subject to several physical effects that can be considered based on experimental correlations. Some of these effects may be lateral or longitudinal oscillations due to the applied take-up system, or to gas flow turbulence in the spinning chamber caused by the gas supply system of the spinning chamber. You can choose to specify a constant for the drag, heat transfer or mass transfer coefficients or use one of the Fluent-provided methods (for example, kase-matsuo) to compute the coefficients. These methods are described below. Alternatively, you can specify a custom drag, heat transfer, or mass transfer coefficient using a user-defined function (UDF). See [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#) for details on how to define and use UDFs in the fiber model.

For more information, see the following sections:

- 2.8.1. Drag Coefficient
- 2.8.2. Heat Transfer Coefficient
- 2.8.3. Mass Transfer Coefficient

## 2.8.1. Drag Coefficient

The following options for drag coefficients are available in the fiber model to compute the drag due to flow moving parallel to the fibers:

### **const-drag**

A constant value for the drag can be specified.

### **kase-matsu**

A drag coefficient using the model taken from Kase and Matsuo [3] (p. 191), see [Equation 2.29 \(p. 149\)](#).

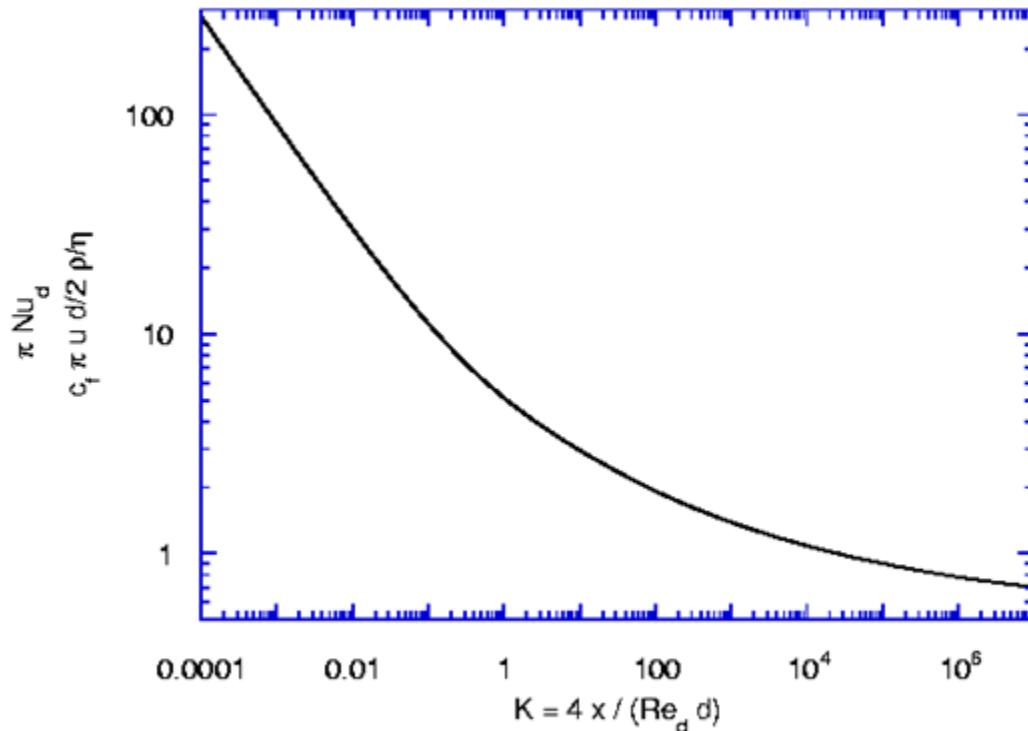
### **gampert**

A drag coefficient using the model from Gampert [2] (p. 191).

Gampert provided analytical and numerical solutions for laminar axisymmetric flow of a moving cylinder in stationary air including strong curvature effects in the boundary layer, [2] (p. 191). The drag coefficient and the Nusselt number are shown as dimensionless groups in [Figure 2.2: Dimensionless Groups of Drag Coefficient and Nusselt Number \(p. 148\)](#).

Note that the curvature  $k$  is defined as the abscissa in [Figure 2.2: Dimensionless Groups of Drag Coefficient and Nusselt Number \(p. 148\)](#). This correlation is recommended in laminar flows.

**Figure 2.2: Dimensionless Groups of Drag Coefficient and Nusselt Number**



**user-defined**

A drag coefficient that you specify in a user-defined function (UDF). See [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#) for more information on using UDFs in the fiber model.

$$C_{f,par} = \frac{1.24}{Re_d^{0.81}} \quad (2.29)$$

In [Figure 2.2: Dimensionless Groups of Drag Coefficient and Nusselt Number \(p. 148\)](#) and [Equation 2.29 \(p. 149\)](#), the Reynolds number is computed based on the relative velocity of the surrounding flow parallel to the fibers  $Re_d = \frac{\rho d(u_f - u_{par})}{\eta}$ .

Lateral drag due to flow of the surrounding fluid perpendicular to the fibers is computed by a correlation from Schlichting [7] (p. 191)

$$C_{f,lat} = 10^{(a_1 + a_2 \log Re_{d,lat} + a_3 \log^2 Re_{d,lat})} \quad (2.30)$$

In [Equation 2.30 \(p. 149\)](#) the Reynolds number is computed based on the relative velocity of the surrounding flow perpendicular to the fibers  $Re_{d,lat} = \frac{\rho d u_{lat}}{\eta}$ .

## 2.8.2. Heat Transfer Coefficient

The following options for heat transfer coefficients are available in the fiber model to compute the exchange of heat between fibers and surrounding flow:

**const-htc**

A constant value for the heat transfer coefficient in SI units can be specified.

**kase-matsuo-1**

A heat transfer coefficient based on a model from Kase and Matsuo [3] (p. 191) that considers pure parallel flow, see [Equation 2.31 \(p. 149\)](#).

$$Nu_d = \frac{\alpha d_f}{\lambda} = 0.42 Re_d^{0.334} \quad (2.31)$$

**kase-matsuo-2**

A heat transfer coefficient based on a model from Kase and Matsuo [3] (p. 191) that also considers cross flow, see [Equation 2.32 \(p. 149\)](#). Refer to [Momentum Exchange \(p. 144\)](#) for definitions of the variables below.

$$Nu_d = \frac{\alpha d_f}{\lambda} = 0.42 Re_d^{0.334} \left[ 1 + \left( \frac{8u_{lat}}{u_f - u_{par}} \right)^2 \right]^{1/6} \quad (2.32)$$

**gampert**

A heat transfer coefficient based on a model from Gampert [2] (p. 191).

Gampert provided analytical and numerical solutions for laminar axisymmetric flow of a moving cylinder in stationary air including strong curvature effects in the boundary layer, [2] (p. 191). The drag coefficient and the Nusselt number are shown as dimensionless groups in [Figure 2.2: Dimensionless Groups of Drag Coefficient and Nusselt Number \(p. 148\)](#). This correlation is recommended for laminar flows.

**user-defined**

A heat transfer coefficient that you specify in a user-defined function (UDF). See [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#) for more information on using UDFs in the fiber model.

## 2.8.3. Mass Transfer Coefficient

The mass transfer coefficients are based on the Reynolds analogy applied to the heat transfer coefficients. The following options are available in the fiber model to compute the exchange of mass between fibers and surrounding flow:

### **const-mtc**

If you select this coefficient, you specify the direct transferred mass flow rate in  $kg/(m^2 s)$ , rather than the mass transfer coefficient.

### **kase-matsuo-1**

A mass transfer coefficient based on a model from Kase and Matsuo [3] (p. 191) that considers pure parallel flow, see [Equation 2.33 \(p. 150\)](#).

$$Sh_d = \frac{\beta d_f}{D_s} = 0.42 Re_d^{0.334} \quad (2.33)$$

### **kase-matsuo-2**

A mass transfer coefficient based on a model from Kase and Matsuo [3] (p. 191) that also considers cross flow, see [Equation 2.34 \(p. 150\)](#).

$$Sh_d = \frac{\beta d_f}{D_s} = 0.42 Re_d^{0.334} \left[ 1 + \left( \frac{8u_{lat}}{u_f - u_{par}} \right)^2 \right]^{1/6} \quad (2.34)$$

### **gampert**

A mass transfer coefficient based on a model from Gampert [2] (p. 191).

Gampert's analytical and numerical solutions for laminar axisymmetric flow of a moving cylinder in stationary air include strong curvature effects in the boundary layer, [2] (p. 191). The Sherwood number is analogous to the Nusselt number as shown as dimensionless groups in [Figure 2.2: Dimensionless Groups of Drag Coefficient and Nusselt Number \(p. 148\)](#). This correlation is recommended for laminar flows.

### **user-defined**

A mass transfer coefficient that you specify in a user-defined function (UDF). See [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#) for more information on using UDFs in the fiber model.

Because these correlations are valid only for nearly-zero mass transfer due to the Reynolds analogy, a film theory is used to compute the nonzero mass transfer, see [Equation 2.2 \(p. 140\)](#).

## 2.9. Fiber Properties

Most properties needed for the fibers can be specified in ANSYS Fluent's material dialog box except the ones described in this section.

### [2.9.1. Fiber Viscosity](#)

### [2.9.2. Vapor-Liquid Equilibrium](#)

### [2.9.3. Latent Heat of Vaporization](#)

### [2.9.4. Emissivity](#)

## 2.9.1. Fiber Viscosity

Fibers are treated as Newtonian fluids. In elongational flow of Newtonian fluids, the elongational viscosity (or sometimes called Trouton viscosity) is related to the zero shear viscosity by a factor of 3. Because

this approach is applied to the computation of melt and dry spun fibers, only the zero shear viscosity is described.

### 2.9.1.1. Melt Spinning

In melt spinning, the fiber is considered to be liquid until its temperature falls below the solidification temperature  $T_{f,solid}$ . For the liquid state an exponential approach is used, see [Equation 2.35 \(p. 151\)](#).

$$\eta_0 = A e^{\left(\frac{B}{C+T_f}\right)} \left( \frac{du_f}{dz} \right)^{(N-1)} \quad (2.35)$$

where  $A$ ,  $B$ ,  $C$ , and  $N$  are user-specified constants.

Below this temperature the value given in the material dialog box for the fiber polymer material is used. Typically, a high value like  $\eta_0 = 1 \times 10^8 \text{ Pas}$  is used for the fiber viscosity to simulate a solid fiber. This value may depend on your polymer and the range of viscosity values in your simulation. You can use every profile available in the materials dialog box except UDF's to describe temperature dependency of the viscosity of the solidified fiber.

The fiber model uses a blending interval for the temperature  $\Delta T_{f,bl} = T_{f,liquid} - T_{f,solid}$  to provide a smooth transition of the viscosity between liquid and solid state of the fiber. The viscosity in this blending interval is computed as

$$\eta_0 = \frac{\Delta T_{f,bl}}{\frac{T_{f,liquid} - T_f}{\eta_{f,solid}(T_f)} + \frac{T_f - T_{f,solid}}{\eta_{f,liquid}(T_f)}} \quad (2.36)$$

#### Important

The chosen values of the blending interval may influence the results. Values for the blending interval should be adapted to the rheological data of the polymer.

### 2.9.1.2. Dry Spinning

In dry spinning, the following approach is used to consider the effect of solvent on the zero shear viscosity:

$$\eta_0 = AB^C \left( \frac{du}{dz} \right)^{(N-1)} (1 - Y_s)^D e^{(E/T_f)} \quad (2.37)$$

In this equation,  $B$  is considered to be the degree of polymerization as it is used by Ohzawa [\[4\] \(p. 191\)](#).  $A$ ,  $B$ ,  $C$ ,  $D$ ,  $E$ , and  $N$  are user-specified constants.

### 2.9.2. Vapor-Liquid Equilibrium

In dry spinning applications, the solvent in the fiber is in thermodynamic equilibrium with the solvent vapor in the surrounding fluid. The saturation vapor pressure of the solvent is computed using an Antoine-type equation

$$p_{s,vap} = e^{\left(A + \frac{B}{T_f + C}\right)} \quad (2.38)$$

If you enable the vapor-liquid equilibrium given by Flory [\[1\] \(p. 191\)](#), [Equation 2.3 \(p. 140\)](#) is used to compute the mole fraction of the solvent in the surrounding gas.

## 2.9.3. Latent Heat of Vaporization

The latent heat of vaporization of the solvent  $\Delta h_s$  has to be specified at the vaporization temperature  $T_{vap}$ . The temperature dependency of the latent heat is not needed because this is treated automatically by the fiber model. Using [Equation 2.16 \(p. 142\)](#) the following equation can be derived to get the latent heat of vaporization at any temperature  $T^*$ :

$$\Delta h_s(T^*) = \Delta h_s(T_{vap}) + \int_{T_{vap}}^{T^*} (C_{p_{sv}} - C_{p_s}) dT \quad (2.39)$$

## 2.9.4. Emissivity

If the P-1 or the discrete ordinates radiation model is enabled, the emissivity of the fiber has to be specified. The fiber diameter can approach the order of magnitude of the wavelength of the irradiation. In this parameter range, additional effects like diffraction take place in addition to scattering and transmission. Because these properties are not well known for fibers, diffraction, scattering, and transmission are neglected. They have to be included only in the fiber emissivity.

## 2.10. Solution Strategies

The fiber model solves sets of differential equations. It provides its own numerical algorithms showing their own numerical difficulties. Therefore it is highly recommended to first achieve a numerical solution for the pure fibers without any coupling to the surrounding flows. If the fiber model does not converge for a given flow field of the surrounding fluid it will not converge for changing fields of the surrounding fluid.

When you start with a fiber simulation, chose the appropriate models needed for the fibers. For the first simulation, disable options like **Include Lateral Drag**, **Fiber Radiation Interaction**, and **Fiber Viscous Heating** to reduce possible interactions.

When specifying the grid of the fibers be sure to refine the grid in the area where large gradients of the velocity appear. This is mainly near the injection point where the fiber is released and near the point of solidification. Because this point is not known a priori, you have to refine the grid during subsequent steps.

If the fiber grid seems to be well suited, you can influence the convergence behavior by starting the iteration with low under-relaxation factors in the **Fiber Solution Controls** dialog box. This may help in most situations where the species and energy equations are strongly coupled (for example, dry spinning applications), or if the solvent has a very high latent heat of vaporization (for example, water).

### Important

Be sure to increase the under-relaxation factor of the momentum equation to 1, when doing a melt spinning case to achieve a converged solution. This should be done after a numerically stable solution has been set up.

When the solution process of the pure fiber equations show a numerically stable behavior, you can increase the complexity of the models by activating viscous heating, or radiation interaction, if such effects are important in their application. After this, you can proceed with a coupled solution by solving the fiber equations and the fluid flow equations alternately. If the solution algorithm of the fluid flow

equations diverges, you must investigate the source terms computed by the fiber model, see [Exchange Terms of Fibers \(p. 185\)](#).

You can damp strong changes of the source terms with a low under-relaxation factor. Another choice is to increase the number of ANSYS Fluent iterations between two subsequent fiber computations.

If the coupled solution process converges, you can increase the under-relaxation factor of the source terms and decrease the number of ANSYS Fluent iterations between two subsequent fiber computations.

You may also want to consider underrelaxing the fluid flow equations. This helps especially for the energy and species equations.



---

## Chapter 3: Using the Continuous Fiber Module

---

The procedure for setting up and solving fiber spinning flows is described in detail in this chapter. Only the steps related to fiber modeling are shown here. Refer to [Continuous Fiber Model Theory \(p. 139\)](#) for information about the theory.

---

### Important

Note that the continuous fiber model is available for the pressure-based solver, only.

---

For information about inputs related to other models used in conjunction with the fiber models, see the appropriate sections for those models in the ANSYS Fluent [User's Guide](#).

- 3.1. [Installing the Continuous Fiber Module](#)
- 3.2. [Loading the Continuous Fiber Module](#)
- 3.3. [Getting Started With the Continuous Fiber Module](#)
- 3.4. [Fiber Models and Options](#)
- 3.5. [Fiber Material Properties](#)
- 3.6. [Defining Fibers](#)
- 3.7. [User-Defined Functions \(UDFs\) for the Continuous Fiber Model](#)
- 3.8. [Fiber Model Solution Controls](#)
- 3.9. [Postprocessing for the Continuous Fibers](#)

### 3.1. Installing the Continuous Fiber Module

The continuous fiber model is provided as an add-on module with the standard ANSYS Fluent licensed software. The module is installed with the standard installation of ANSYS Fluent in a directory called `addons/fiber` in your installation area. The continuous fiber module consists of a UDF library and a pre-compiled scheme library, which must be loaded and activated before calculations can be performed.

### 3.2. Loading the Continuous Fiber Module

The continuous fiber module is loaded into ANSYS Fluent through the text user interface (TUI). The module can be loaded only when a valid ANSYS Fluent case file has been set or read. The text command to load the module is

```
define → models → addon-module
```

A list of ANSYS Fluent add-on modules is displayed:

```
> /define/models/addon-module
Fluent  Addon Modules:
 0. None
 1. MHD Model
 2. Fiber Model
 3. Fuel Cell and Electrolysis Model
 4. SOFC Model with Unresolved Electrolyte
 5. Population Balance Model
 6. Adjoint Solver
 7. Battery Module
 8. MSMD Battery Module
```

```
9. PEM Fuel Cell Model  
Enter Module Number: [0] 2
```

Select the continuous fiber model by entering the module number 2. During the loading process a scheme library containing the graphical and text user interface, and a UDF library containing a set of user-defined functions (UDFs) are loaded into ANSYS Fluent.

During this process, you will be asked the question

```
Preset all fiber model specific UDF hooks? [no]
```

If you answer *yes* the standard fiber source term UDFs will be assigned to *all* fluid zones in your case, and a message will be reported to the console window confirming this:

```
Assigning standard fiber source terms to all fluid zones.
```

If you answer *no* to presetting source term UDFs to all fluid zones in the domain, then three options will be available to you when setting up source terms for fluid zones in your fiber model: no source, constant source, or UDF source. Note that it is your responsibility to specify the rest of the settings for a proper fiber simulation. See [Source Term UDF Setup \(p. 158\)](#) for details.

If you are loading an existing case file then you should answer the question with *no*. Otherwise, your saved source term settings will be replaced by a UDF.

If a mixture material has been defined, then you will be asked an additional question

```
Preset mass exchange source terms hooks? [no]
```

If you intend to conduct a dry spinning simulation, then you should reply *yes*.

During the loading process the UDF library for the continuous fiber module is loaded in ANSYS Fluent. This is reported to the console (see below). The UDF library also becomes visible as a new entry in the **UDF Library Manager** dialog box. The basic setup of the continuous fiber model is performed automatically when the fiber module is successfully loaded.

```
Opening library "/.../addons/fiber"...
Library "/...addons/fiber/lnx86/3d/libudf.so" opened
fm_adjust
fm_src_mass
fm_src_x_mom
fm_src_y_mom
fm_src_z_mom
fm_src_enthalpy
fm_src_dom
fm_on_demand Done.

Addon Module: fiber...loaded!
```

If you did not preset the fiber model specific UDF hooks, you will need to check allocation of user-defined memory, hook an adjust function (*fm\_adjust*) to ANSYS Fluent, and set up the source terms on your own. This is explained in [User-Defined Memory and the Adjust Function Setup \(p. 157\)](#).

---

### Important

Note that user-defined memory locations for the fiber model will not be allocated properly if you do not initialize the flow field. If you are setting up a fiber computation based on a converged case, you *must* re-load the ANSYS Fluent data file after initializing the solution.

---

The continuous fiber module setup is saved with the ANSYS Fluent case file. The module is loaded automatically when the case file is subsequently read into ANSYS Fluent. Note that in the saved case file, the continuous fiber module is saved with the absolute path. Therefore, if the locations of the continuous fiber module installation or the saved case file are changed, then ANSYS Fluent will not be able to load the module when the case file is subsequently read. In this situation, you will have to unload the UDF library using the **UDF Library Manager** dialog box after the case file is read, and then reload the continuous fiber module. To unload the UDF library go to the **UDF Library Manager** dialog box



**User-Defined** → **User-Defined** → **Functions** → **Manage...**

select the fiber library under **UDF Libraries**, and click **Unload**. Previously-saved continuous fiber model setup and parameters will be preserved in this process.

### 3.3. Getting Started With the Continuous Fiber Module

The continuous fiber model is implemented by user-defined functions (UDFs) and scheme routines in ANSYS Fluent. A number of UDFs are used to solve the fiber equations. When you loaded the fiber module in the previous step ([Loading the Continuous Fiber Module \(p. 155\)](#)), UDF and scheme libraries that are required by the continuous fiber model were *automatically* loaded. Before you can begin the process of defining your fiber model, however, you will need to perform some additional setup tasks that involve allocating user-defined memory for the UDFs and hooking an adjust UDF to ANSYS Fluent. Follow the procedure below.

For more information, see the following sections:

- [3.3.1. User-Defined Memory and the Adjust Function Setup](#)
- [3.3.2. Source Term UDF Setup](#)

#### 3.3.1. User-Defined Memory and the Adjust Function Setup

1. Allocate user-defined memory for the model by incrementing the **Number of User-Defined Memory Location** to 8 in the **User-Defined Memory** dialog box.



##### Important

Note that you *must* initialize your solution (in the **Solution Initialization** task page) in order for user-defined memory to be allocated properly. If you are setting up a fiber simulation based on a converged case, then you will have to reload the ANSYS Fluent data file after initializing the solution.

2. Hook the adjust function UDF to ANSYS Fluent by choosing `fm_adjust::fiber` from the drop-down list for **Adjust** in the **User-Defined Function Hooks** dialog box.



**User-Defined** → **User-Defined** → **Function Hooks...**

### 3.3.2. Source Term UDF Setup

If you answered no to presetting all fiber model-specific UDF hooks during the loading process ([Loading the Continuous Fiber Module \(p. 155\)](#)) then you will need to set source terms, individually, for each fluid zone in your model. Alternatively, you can leave the default settings (none for no source term).

For each fluid zone in your model, specify none, constant, or UDF for all of the source terms by following the procedure below:

 **Setup** →  **Cell Zone Conditions**

1. In the **Cell Zone Conditions** task page, select a fluid zone under **Zone** and click **edit....** This opens the **Fluid** dialog box.
2. In the **Fluid** dialog box, check **Source Terms** and click the **Source Terms** tab.
3. For each of the source terms in the scroll list (**Mass**, **X Momentum**, and so on), click the **Edit...** button next to each source term to open the corresponding source dialog box. Leave the default none, or choose constant or UDF from the drop-down list. Choose the UDF in the drop-down list that corresponds to the particular source term. For example, `udf fm_src_mass` corresponds to the Mass source term. Use the table below ([Table 3.1: Source Terms and Corresponding UDFs \(p. 158\)](#)) as a reference guide.

**Table 3.1: Source Terms and Corresponding UDFs**

<b>Mass</b>	<code>udf fm_src_mass</code>
<b>X Momentum or Axial Momentum</b>	<code>udf fm_src_x_mom</code>
<b>Y Momentum or Radial Momentum</b>	<code>udf fm_src_y_mom</code>
<b>Z Momentum</b>	<code>udf fm_src_z_mom</code>
<b>Energy</b>	<code>udf fm_src_enthalpy</code>
<b>discrete ordinates model</b>	<code>udf fm_src_dom</code>

4. Click **OK** when all of the UDFs have been hooked.
5. Repeat this process for the remaining fluid zones in your ANSYS Fluent model.

#### Important

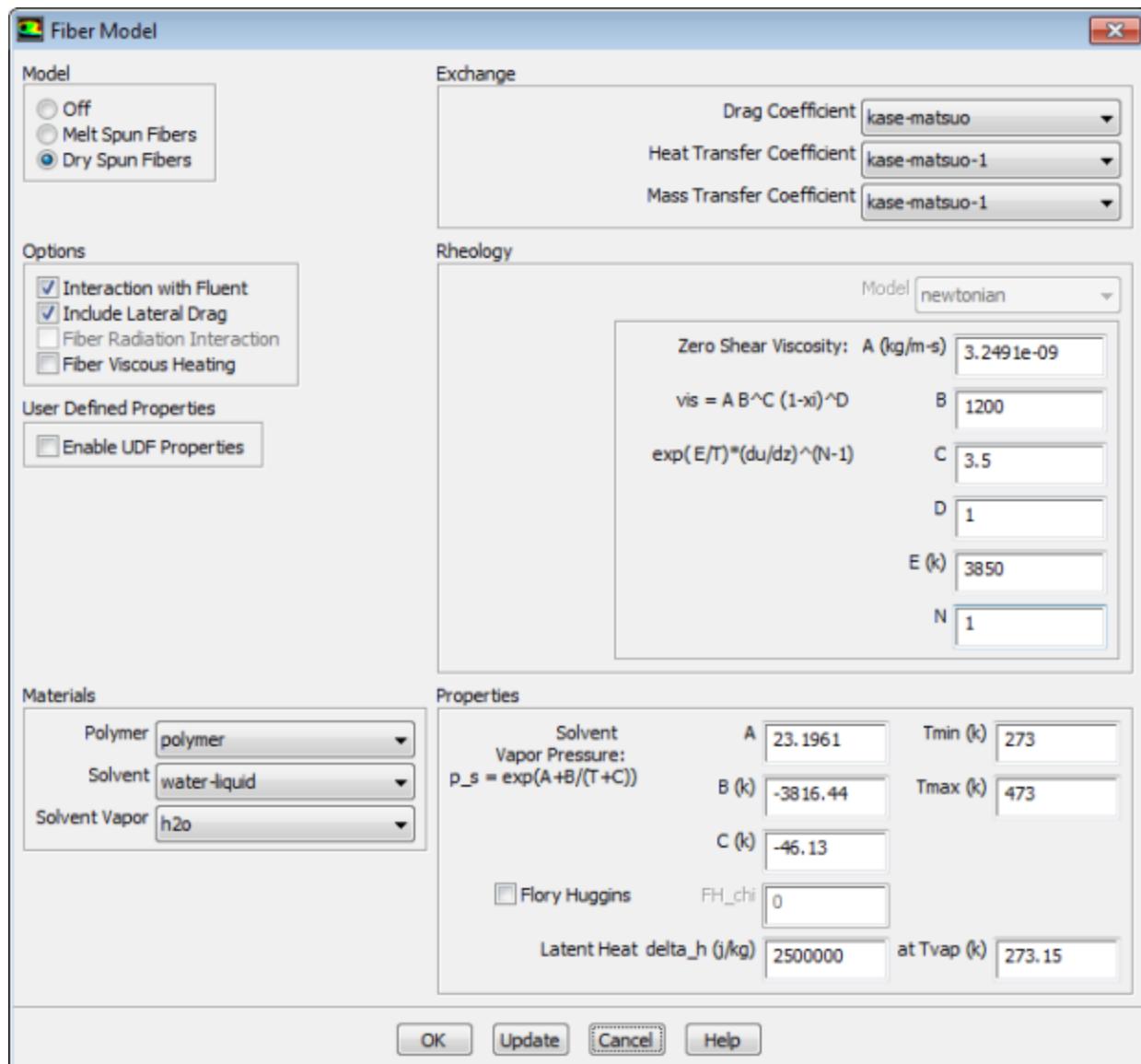
If you want to include radiative interaction of the fibers with the discrete ordinate (DO) radiation model, then the appropriate source term UDF (`udf fm_src_dom`) will be hooked *automatically* when you select **Fiber Radiation Interaction** in the **Fiber Model** dialog box. You *must* initialize the solution (which will allocate memory for the DO model) before the fiber model will be ready to accept the fiber radiation interaction data.

### 3.4. Fiber Models and Options

This section provides information on how to choose the type of fiber model you want to implement as well as select various model options that are available in ANSYS Fluent's fiber model. The fiber **Model** and **Options** are selected in the **Fiber Model** dialog box ([Figure 3.1: The Fiber Model Dialog Box \(p. 159\)](#)).

Setup → Models → Continuous Fiber Spinning Edit...

Figure 3.1: The Fiber Model Dialog Box



For more information, see the following sections:

- 3.4.1. Choosing a Fiber Model
- 3.4.2. Including Interaction With Surrounding Flow
- 3.4.3. Including Lateral Drag on Surrounding Flow
- 3.4.4. Including Fiber Radiation Interaction
- 3.4.5. Viscous Heating of Fibers
- 3.4.6. Drag, Heat and Mass Transfer Correlations

### 3.4.1. Choosing a Fiber Model

With ANSYS Fluent's fiber model, you can model two types of fibers: melt spun and dry spun. The model you choose depends on the polymer that is used to draw the fibers. If the fiber polymer can be molten without being destroyed thermally, then a melt spun fiber process is typically used to produce

the fibers. In such cases, select **Melt Spun Fibers** under **Model** in the **Fiber Model** dialog box (Figure 3.1: The Fiber Model Dialog Box (p. 159)).

When the fiber polymer can be liquefied with a suitable solvent, or the fiber polymer's production process involves a solvent, the fibers are formed typically in a dry spinning process. In such cases, select **Dry Spun Fibers** under **Model** in the **Fiber Model** dialog box (Figure 3.1: The Fiber Model Dialog Box (p. 159)).

### 3.4.2. Including Interaction With Surrounding Flow

If the fibers in your simulation strongly influence the flow of the surrounding fluid and need to be considered, you must select **Interaction with Fluent** under **Options** in the **Fiber Model** dialog box (Figure 3.1: The Fiber Model Dialog Box (p. 159)). When iterating a solution with ANSYS Fluent, the fiber model equations are solved alternating with ANSYS Fluent's flow equations. Source terms are also computed to couple the fiber equations with the fluid flow equations. The calculation of the source terms is performed only during the course of an ANSYS Fluent computation. See [Coupling Between Fibers and the Surrounding Fluid \(p. 144\)](#) for a description of the source terms.

### 3.4.3. Including Lateral Drag on Surrounding Flow

In typical fiber simulations, the axial drag of the fibers is the most important force acting on the fibers as well as on the surrounding fluid. In some situations, the lateral or cross-flow drag can become important. This is the case when the fibers are mainly cooled or dried in a cross-flow situation. In such cases you can select **Include Lateral Drag** under **Options** in the **Fiber Model** dialog box (Figure 3.1: The Fiber Model Dialog Box (p. 159)). The drag is estimated for a cylinder in cross-flow and is shown in [Equation 2.30 \(p. 149\)](#). Lateral drag is not considered by the fiber equations and therefore lateral bending of the fibers is not considered.

### 3.4.4. Including Fiber Radiation Interaction

In some situations radiative heat exchange is important. If you are using ANSYS Fluent's P-1 radiation model or ANSYS Fluent's discrete ordinates (DO) radiation model, you can consider the effects of irradiation on the cooling and heating of the fibers. When you are using the DO radiation model, the effects of the fibers upon the DO model is considered as well. If the DO radiation model is enabled, the **Fiber Radiation Interaction** option will be turned on and you will need to enter the fiber's **Emissivity** under **Properties** in the **Fiber Model** dialog box (Figure 3.1: The Fiber Model Dialog Box (p. 159)).

### 3.4.5. Viscous Heating of Fibers

When high elongational drawing rates are combined with large fiber viscosities, viscous heating of fibers may become important. To consider this effect in the fiber energy equation (Equation 2.12 (p. 141)), you can select **Fiber Viscous Heating** under **Options** in the **Fiber Model** dialog box (Figure 3.1: The Fiber Model Dialog Box (p. 159)).

### 3.4.6. Drag, Heat and Mass Transfer Correlations

The effects of the boundary layer of the fiber are modeled in terms of drag, heat transfer, and mass transfer coefficients. These parameters are specified under **Exchange** in the **Fiber Model** dialog box (Figure 3.1: The Fiber Model Dialog Box (p. 159)).

- For the **Drag Coefficient**, you can choose between **const-drag**, **kase-matsu**, **gampert** and **user-defined** from the drop-down list. If you choose **const-drag**, the constant you enter must be specified as a dimension-

less value. See [Drag Coefficient \(p. 148\)](#) for a description of these options. User-defined functions (UDFs) are described in detail in [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#).

- For the **Heat Transfer Coefficient** you can choose between **const-htc**, **kase-matsuo-1**, **kase-matsuo-2**, **gampert**, and **user-defined** from the drop-down list. If you choose **const-htc**, the constant you enter must be specified SI units of  $W / (m^2 K)$ . See [Heat Transfer Coefficient \(p. 149\)](#) for a description of these options. User-defined functions (UDFs) are described in detail in [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#).
- For the **Mass Transfer Coefficient** you can choose between **const-mtc**, **kase-matsuo-1**, **kase-matsuo-2**, **gampert**, and **user-defined**. If you choose **const-mtc**, you must enter the mass transfer rate in units of  $kg / (m^2 s)$  instead of the mass transfer coefficient. See [Mass Transfer Coefficient \(p. 150\)](#) for a description of these options. User-defined functions (UDFs) are described in detail in [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#).

## 3.5. Fiber Material Properties

For more information, see the following sections:

[3.5.1. The Concept of Fiber Materials](#)

[3.5.2. Description of Fiber Properties](#)

### 3.5.1. The Concept of Fiber Materials

The material properties you specify for the fibers are used for all fibers defined in your model. You cannot consider fibers consisting of different fiber materials in one simulation.

The continuous fiber model makes use of ANSYS Fluent's material concept for the **Material Type** of the **fluid**. Because not all properties are available in ANSYS Fluent's **Create/Edit Materials** dialog box for this material type, some additional property information can be provided through the **Fiber Model** dialog box ([Figure 3.1: The Fiber Model Dialog Box \(p. 159\)](#)). The procedure to define the material properties for the fibers in your simulation is as follows:

- In the **Create/Edit Materials** dialog box, set the **Material Type** as **fluid**. This fluid will be used as the polymer or solvent in the fiber.
- Enter all data for this material.

You can use all profiles available in the **Create/Edit Materials** dialog box to define the properties as functions of temperature. In order to invoke user-defined fiber properties, you need to use the UDF template file (see [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#) for details). UDF access is available for viscosity, density, specific heat capacity, thermal conductivity and solvent liquid-vapor equilibrium pressure.

- Select the **Polymer** in the **Materials** group box of the **Fiber Model** dialog box.
- For dry spinning simulations, you also have to select **Solvent** and the gas phase species that represents the **Solvent Vapor**.
- Enter any additional data needed for the fiber material in the **Fiber Model** dialog box.

### 3.5.2. Description of Fiber Properties

The properties that appear in the **Fiber Model** dialog box vary depending on the fiber model type.

The following list describes the properties you may need for a fiber material. For every property listed, the dialog box name is provided where the property can be defined.

### Blending Interval

(**Fiber Model** dialog box) is the temperature interval used to compute an average of the fiber viscosities in liquid and solid state of **Melt Spun Fibers**. This option is only available when the **Melt Spun Fibers** option is selected. See [Fiber Viscosity \(p. 150\)](#) for details about how the **Blending Interval** is applied to the fiber viscosity.

### C<sub>p</sub>

(**Create/Edit Materials** dialog box) is the specific heat,  $C_{p,f}$ , of the fiber in units of energy per mass and temperature. In the case of dry spun fibers, a mass average is computed based on the values entered for **Polymer** and its **Solvent**. You can use any of the functions available to define temperature dependency. If you want to use a user-defined function profile, you need to modify the UDF template provided by ANSYS Fluent. See [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#) for details.

### Density

(**Create/Edit Materials** dialog box) is the density,  $\rho_f$ , of the fiber in units of mass per unit volume. This density is the mass density and not the volume density. In the case of dry spun fibers, a mass average is computed based on the values entered for **Polymer** and its **Solvent**. You can use any of the functions available to define temperature dependency. If you want to use a user-defined function profile, you need to modify the UDF template provided by ANSYS Fluent. See [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#) for details.

### Emissivity

(**Fiber Model** dialog box) is the emissivity of fibers in your model,  $\varepsilon_f$ , used to compute radiation heat transfer to the fibers ([Equation 2.13 \(p. 141\)](#), [Equation 2.14 \(p. 141\)](#), [Equation 2.24 \(p. 146\)](#) and [Equation 2.25 \(p. 146\)](#)) when the P-1 or discrete ordinates radiation model is active. Note that you must enable radiation to fiber, using the **Fiber Radiation Interaction** option in the **Fiber Model** dialog box.

### Flory Huggins

(**Fiber Model** dialog box) can be enabled to apply [Equation 2.3 \(p. 140\)](#) to compute the vapor-liquid equilibrium at the fiber surface. When it is enabled, you have to specify an appropriate value for the dimensionless Flory Huggins parameter,  $\chi$ . This option is only visible when **Dry Spun Fibers** has been chosen.

### Latent Heat

(**Fiber Model** dialog box) is the latent heat of vaporization of the **Solvent** when evaporated from a dry spun fiber. Note that you have to enter the vaporization or reference temperature,  $T_{vap}$ , where the specified value of the latent heat has been measured. This vaporization temperature is used to automatically consider the change of latent heat with temperature. See [Equation 2.16 \(p. 142\)](#) and [Equation 2.39 \(p. 152\)](#) for more information on how this is achieved. These options are only visible when **Dry Spun Fibers** has been chosen.

### Solidification Temperature

(**Fiber Model** dialog box) is the temperature below which the fiber polymer of a **Melt Spun Fiber** will solidify. It will be used when computing the viscosity of **Melt Spun Fibers**. This option is only visible when **Melt Spun Fibers** has been chosen.

### Solvent Vapor Pressure

(**Fiber Model** dialog box) is the vapor pressure of the solvent evaporating from the fiber surface in dry spinning. You have to enter coefficients for an Antoine-type equation ([Equation 2.38 \(p. 151\)](#)). Note that the coefficients must be entered in such units that the outcome of the Antoine-type equation is in Pascal. In addition to the coefficients of the Antoine equation, you have to enter the range of validity for the vapor

pressure. Below the minimal temperature the vapor pressure at the minimal temperature will be used. Above the maximal temperature, the vapor pressure at the maximal temperature is used.

## Thermal Conductivity

(**Create/Edit Materials** dialog box) is the thermal conductivity,  $\lambda_f$ , of the fiber in units of power per length and temperature. In the case of dry spun fibers, a mass average is computed based on the values entered for **Polymer** and its **Solvent**. You can use any of the functions available to define temperature dependency. If you want to use a user-defined function profile, you need to modify the UDF template provided by ANSYS Fluent. See [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#) for details.

## Zero Shear Viscosity

(**Create/Edit Materials** dialog box and/or the **Fiber Model** dialog box, depending on the chosen fiber model) is the fiber viscosity,  $\eta_f$ , at zero shear rate. It is not the elongational or Trouton viscosity.

For **Melt Spun Fibers**, you have to enter the viscosity of the fiber in solid state in the **Create/Edit Materials** dialog box for the fluid you have selected as the fiber polymer. Typically this value will be very high compared to the liquid fiber viscosity to represent the fibers as solids. For the solid fiber viscosity you can make use of any temperature-dependent function available in the **Create/Edit Materials** dialog box. If you want to use a user-defined function profile, you need to modify the UDF template provided by ANSYS Fluent. See [User-Defined Functions \(UDFs\) for the Continuous Fiber Model \(p. 175\)](#) for details.

In the **Fiber Model** dialog box, you have to enter the coefficients for the fiber viscosity in liquid state ([Equation 2.35 \(p. 151\)](#)). To define viscosity as a function of fiber velocity gradient, set  $N$  to a value different than 1. In the case of **Melt Spun Fibers**, you also have to enter data for the **Solidification Temperature** and the **Blending Interval**. The blending of the viscosities in liquid and solid state will be computed based on [Equation 2.36 \(p. 151\)](#).

For **Dry Spun Fibers** you only have to enter the coefficients for [Equation 2.37 \(p. 151\)](#) in the **Fiber Model** dialog box. To specify fiber viscosity as a function of fiber velocity gradient, set  $N$  to a value different than 1. Any value entered in the **Create/Edit Materials** dialog box for viscosities of the fluids used as fiber polymer and fiber solvent will not be considered.

---

### Note

Note that depending on fiber velocity gradient, the numerical behavior of the Fiber Model equations may become unstable in combination with fiber viscosity.

---

## 3.6. Defining Fibers

For more information, see the following sections:

- 3.6.1. Overview
- 3.6.2. Fiber Injection Types
- 3.6.3. Working with Fiber Injections
- 3.6.4. Defining Fiber Injection Properties
- 3.6.5. Point Properties Specific to Single Fiber Injections
- 3.6.6. Point Properties Specific to Line Fiber Injections
- 3.6.7. Point Properties Specific to Matrix Fiber Injections
- 3.6.8. Define Fiber Grids

### 3.6.1. Overview

The primary inputs that you must provide for the continuous fiber model calculations in ANSYS Fluent are the starting positions, mass flow rate, take up positions, and other parameters for each fiber. These provide the boundary conditions for all dependent variables to be solved in the continuous fiber model. The primary inputs are:

- Start point ( $x, y, z$  coordinates) of the fiber.
- Number of fibers in group. Each defined fiber can represent a group of fibers that will be used only to compute the appropriate source terms of a group of fibers.
- Diameter of the fiber nozzle,  $d_f$ .
- Mass flow rate per nozzle to compute the velocity of the fiber fluid in the nozzle. The velocity is used as boundary condition for the fiber momentum equation.
- Temperature of the fiber at the nozzle,  $T_f$ .
- Solvent mass fraction of the fiber fluid in the nozzle. This value is used as boundary condition for the solvent continuity equation.
- Take-up point ( $x, y, z$  coordinates) of the fiber.
- Velocity or force at take-up point to describe the second boundary condition needed for the fiber momentum equation (see [Equation 2.4 \(p. 140\)](#)).

In addition to these parameters, you have to define parameters for the grid that is distributed between the start position and take-up point of the fibers. On this grid the dependent fiber variables are solved, by discretizing [Equation 2.1 \(p. 139\)](#), [Equation 2.4 \(p. 140\)](#), and [Equation 2.11 \(p. 141\)](#).

You can define any number of different sets of fibers provided that your computer has sufficient memory.

### 3.6.2. Fiber Injection Types

You will define boundary conditions and grids for a fiber by creating a fiber “injection” and assigning parameters to it. In the continuous fiber model the following types of injections are provided:

- **single**

Use this option to define a single fiber.

- **line**

Use this option when the fibers you want to define start from a line and the starting points are located at constant intervals on this line.

- **matrix** (only in 3D)

Use this option when the fiber starting points are arranged in the shape of a rectangle.

- **file**

Use an ASCII file for entering coordinates and material properties of individual fiber injections in a tabular format as shown below.

```

; 1st commentary line
; 2nd commentary line
; m-th commentary line
x1   Y1   z1   df1   m1   Tf1   Ys1
x2   Y2   z2   df2   m2   Tf2   Ys2
...
xn   Yn   zn   dfn   mn   Tfn   Ysn

```

where each injection is described by the following data fields, all on one line, separated by spaces:

x, y, and z are the Cartesian coordinates of the injection point. The z coordinate must be included for both 3D and 2D solvers to maintain the sequence in which the data entries are interpreted, however, the z coordinate is not used in 2D.

d<sub>f</sub> is the fiber diameter

m is the mass flow rate

T<sub>f</sub> is the temperature

Y<sub>s</sub> is the solvent liquid mass fraction (ignored for melt spinning)

You can include an arbitrary number of comments anywhere in the file. Commentary lines must begin with a semicolon (;).

### 3.6.3. Working with Fiber Injections

**Figure 3.2: The Fiber Injections Dialog Box**



You will use the **Fiber Injections** dialog box (Figure 3.2: The Fiber Injections Dialog Box (p. 165)) to create, modify, copy, delete, initialize, compute, print, read, write, and list fiber injections. To access the **Fiber Injections** dialog box, first make sure you enable a fiber model, then go to

**Setup** → **Models** → **Fiber-Injections** **Edit...**

### 3.6.3.1. Creating Fiber Injections

To create a fiber injection, click **Create**. A new fiber injection appears in the **Fiber Injections** list and the **Set Fiber Injection Properties** dialog box opens automatically to enable you to specify the fiber injection properties (as described in [Point Properties Specific to Single Fiber Injections \(p. 172\)](#)).

### 3.6.3.2. Modifying Fiber Injections

To modify an existing fiber injection, select its name in the **Fiber Injections** list and click **Set....** The **Set Fiber Injection Properties** dialog box opens, and you can modify the properties as needed.

### 3.6.3.3. Copying Fiber Injections

To copy an existing fiber injection to a new fiber injection, select the existing injection in the **Fiber Injections** list and click **Copy**. The **Set Fiber Injection Properties** dialog box will open with a new fiber injection that has the same properties as the fiber injection you have selected. This is useful if you want to set another injection with similar properties.

### 3.6.3.4. Deleting Fiber Injections

You can delete a fiber injection by selecting its name in the **Fiber Injections** list and clicking **Delete**.

### 3.6.3.5. Initializing Fiber Injections

To initialize all solution variables of the fibers defined in a fiber injection, select its name in the **Fiber Injections** list and click **Initialize**. The solution variables will be set to the boundary condition values at the starting points of the fibers.

You can select several fiber injections when you want to initialize several fiber injections at one time.

---

#### Important

If you do not select a fiber injection and click **Initialize**, all fiber injections will be initialized.

---

### 3.6.3.6. Computing Fiber Injections

To solve the fiber equations of a fiber injection for a number of iterations, select its name in the **Fiber Injections** list and click **Compute**. The solution variables of the fibers defined in this fiber injection will be updated for the number of iterations specified in [Figure 3.13: Fiber Solution Controls Dialog Box \(p. 182\)](#).

You can select several fiber injections when you want to compute several fiber injections at one time.

---

#### Important

If you do not select a fiber injection and click **Compute**, all fiber injections will be computed.

---

### 3.6.3.7. Print Fiber Injections

To print the fiber solution variables of a fiber injection into a file, select its name in the **Fiber Injections** list and click **Print**. A file name is generated automatically based on the name of the fiber injection and the number of the fiber. The solution variables of each fiber is stored in a separate file. You may use this file for an external postprocessing or analysis of fiber data.

You can select several fiber injections when you want to print several fiber injections at one time.

---

### **Important**

If you do not select a fiber injection and click **Print**, all fiber injections will be printed.

---

### **3.6.3.8. Read Data of Fiber Injections**

To read the data (properties and solution variables) of a fiber injection previously stored in a file, click **Read Data**. A file selection dialog box is opened where you can select the name of the file in a list or you can enter the file name directly. The names of all fiber injections included in the file are compared with the fiber injections already defined in the model. If the fiber injection exists already in your model, you are asked to overwrite it.

### **3.6.3.9. Write Data of Fiber Injections**

While the settings of the continuous fiber model for numerics and models are stored in the ANSYS Fluent case file, the data of the fibers and defined injections have to be stored in a separate file. To write the data of a fiber injection to a file, click **Write Data**. A file selection dialog box is opened where you can select the name of an existing file to overwrite it or you can enter the name of a new file.

You can select several fiber injections when you want to store several fiber injections in one file.

---

### **Important**

If you do not select a fiber injection and click **Write Data**, all fiber injections will be stored in the specified file.

---

### **3.6.3.10. Write Binary Data of Fiber Injections**

To write the data of a fiber injection in binary format to a file, click **Write Binary Data**. A file selector dialog box is opened where you can select the name of an existing file to overwrite it or you can enter the name of a new file.

You can select several fiber injections when you want to store several fiber injections in binary format in one file.

---

### **Important**

If you do not select a fiber injection and click **Write Binary Data**, all fiber injections will be stored in binary format in the specified file.

---

### **3.6.3.11. List Fiber Injections**

To list starting positions and boundary conditions of the fibers defined in a fiber injection, click **List**. ANSYS Fluent displays a list in the console window. For each fiber you have defined, the list contains the following (in SI units):

- File number in the injection in the column headed (**NO**).
- $x$ ,  $y$ , and  $z$  position of the starting point in the columns headed (**X**), (**Y**), and (**Z**).

- Fiber velocity in the column headed ( $U$ ).
- Temperature in the column headed ( $T$ ).
- Solvent mass fraction in the column headed (SOLVENT).
- Diameter in the column headed (DIAM).
- Mass flow rate in the column headed (MFLOW).
- The number of fibers represented by this fiber group in the column headed (FIBERS).
- The number of fiber grid cells defined for this fiber in the column headed (CELLS).
- A notification whether the starting point of the fiber is located inside or outside the domain in the column headed (IN DOMAIN?).

The boundary conditions at the take up point are also listed. This list consists of the following (in SI units):

- $x$ ,  $y$ , and  $z$  position of the take-up point in the columns headed ( $X$ ), ( $Y$ ), and ( $Z$ ).
- Boundary condition type and its specified value (VELOCITY for given take-up velocity, FORCE for given force in the fiber) in the column headed (BOUNDARY CONDITION).

You can select several fiber injections when you want to list several fiber injections.

---

### **Important**

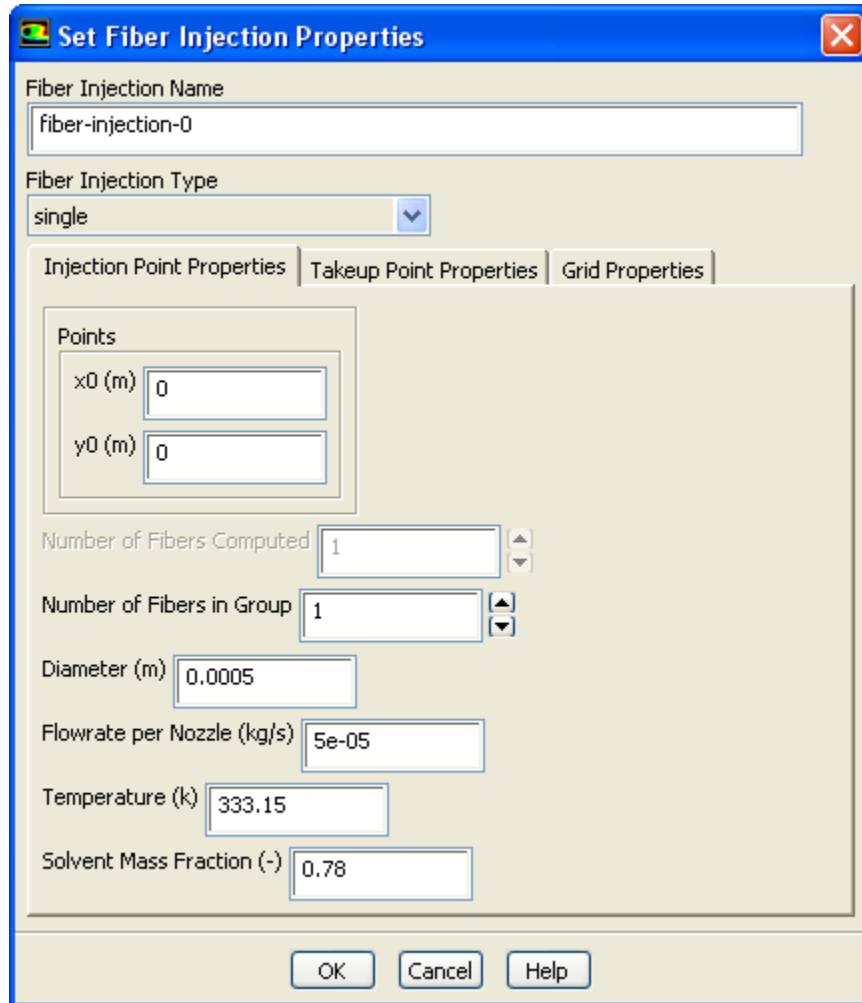
If you do not select a fiber injection and click **List**, all fiber injections will be listed.

---

### **3.6.4. Defining Fiber Injection Properties**

Once you have created an injection (using the **Fiber Injections** dialog box, as described in [Creating Fiber Injections \(p. 166\)](#)), you will use the **Set Fiber Injection Properties** dialog box ([Figure 3.3: The Set Fiber Injection Properties Dialog Box \(p. 169\)](#)) to define the fiber injection properties. (Remember that this dialog box opens when you create a new fiber injection, or when you select an existing fiber injection and click **Set...** in the **Fiber Injections** dialog box.)

**Figure 3.3: The Set Fiber Injection Properties Dialog Box**



The procedure for defining a fiber injection is as follows:

1. If you want to change the name of the fiber injection from its default name, enter a new one in the **Fiber Injection Name** field. This is recommended if you are defining a large number of injections so you can easily distinguish them.
2. Choose the type of fiber injection in the **Fiber Injection Type** drop-down list. The three choices (**single**, **line**, and **matrix**) are described in [Fiber Injection Types \(p. 164\)](#).
3. Click the **Injection Point Properties** tab (the default), and specify the point coordinates according to the fiber injection type, as described in [Point Properties Specific to Single Fiber Injections \(p. 172\)](#)–[Point Properties Specific to Matrix Fiber Injections \(p. 172\)](#).
4. If each of the defined fibers is referring to a group of fibers, enter the number of fibers in **Number of Fibers in Group**. If your nozzle plate has 400 holes and you can simulate them as a line fiber injection with 5 groups, you have to enter a value of 80. This means that only 5 fibers are solved numerically, but each of these fibers stands for 80 fibers to be used to compute the source terms for the surrounding fluid.

This enables you to reduce computing efforts while achieving a proper coupling with the surrounding fluid.

---

**Important**

Note that the **Number of Fibers in Group** is applied to all fibers, defined in your injection. If the number of fiber groups in your line or matrix injection is not the same for all fibers in this injection, you should split this injection into several fiber injections.

---

5. Specify the diameter of the nozzle in the **Diameter** field.
  6. Enter the mass flow rate for a single nozzle in the **Flow rate per Nozzle** field. This will be used to compute the starting velocity of the fiber fluid.
- 

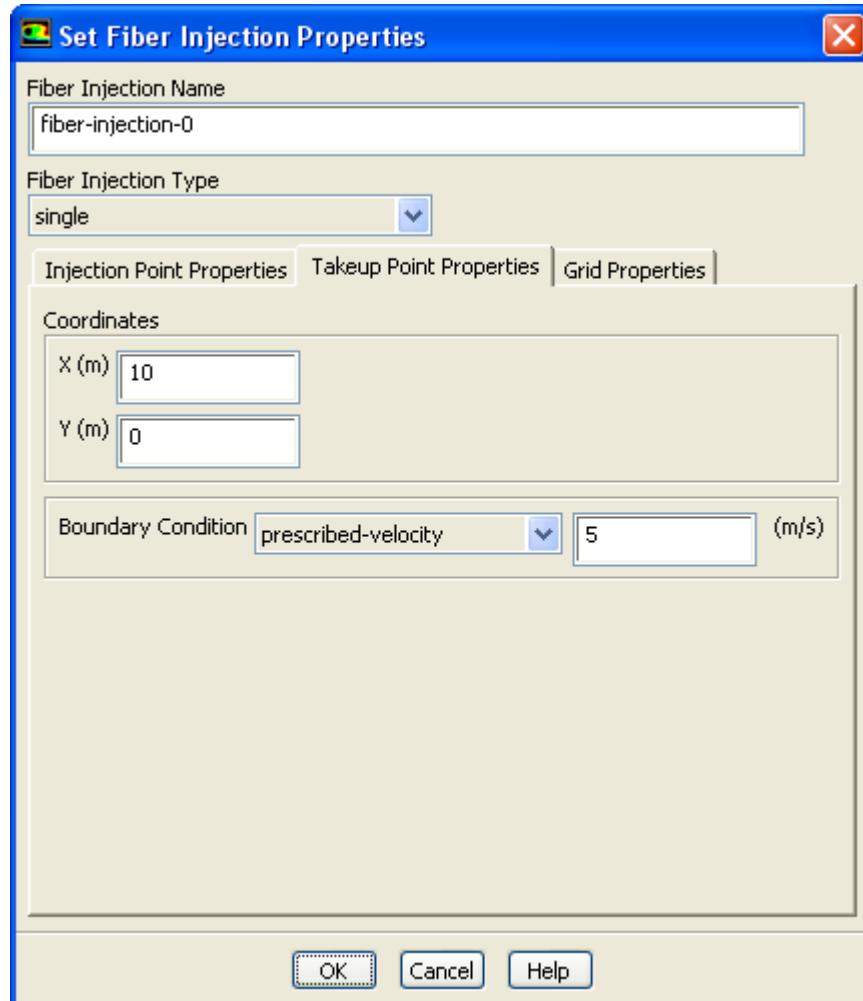
**Important**

Note that the value specified refers to one single nozzle and not to the mass flow rate of all fibers defined in this fiber injection.

---

7. Specify the temperature of the fiber fluid leaving the nozzle in the **Temperature** field.
8. If you are modeling dry spun fibers you also have to enter the solvent's mass fraction at the nozzle in the **Solvent Mass Fraction** field.

**Figure 3.4: The Set Fiber Injection Properties Dialog Box With Take-Up Point Properties**



- Click the **Takeup Point Properties** tab and enter the coordinates of the take-up point (see [Figure 3.4: The Set Fiber Injection Properties Dialog Box With Take-Up Point Properties \(p. 171\)](#)).

### Important

Note that all fibers defined in the fiber injection are collected at the same point. If the fibers of your line or matrix injection vary in this property, you have to define them using several fiber injections.

- Select the appropriate boundary condition from the **Boundary Condition** drop-down list and specify the value for this boundary condition. Choose **prescribed-velocity** if you know the drawing or take-up velocity. Choose **tensile-force** if you want to prescribe a given tensile force in the fiber at the take-up point.
- Click the **Grid Properties** tab and enter all data needed to generate the fiber grid as described in [Define Fiber Grids \(p. 173\)](#).

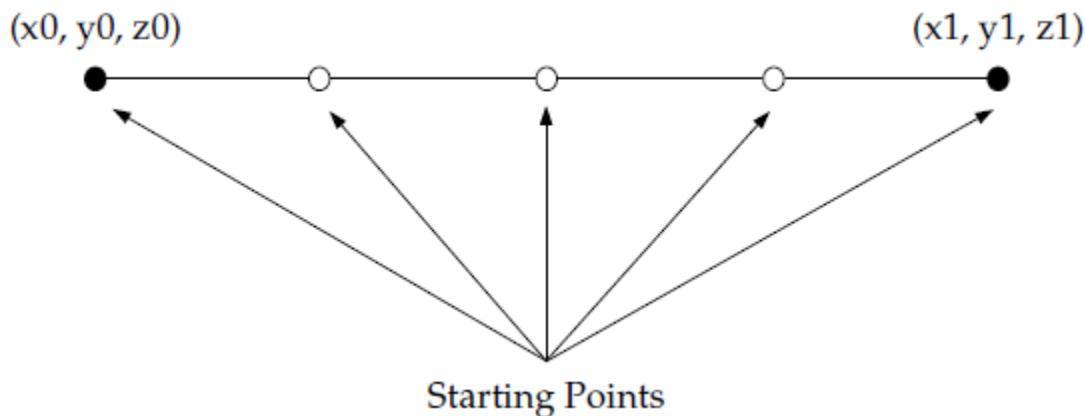
### 3.6.5. Point Properties Specific to Single Fiber Injections

For a single fiber injection, you have to specify the coordinates of the starting point of the fiber. Click the **Injection Point Properties** tab and set the  $x$ ,  $y$ , and  $z$  coordinates in the  **$x0$ ,  $y0$ , and  $z0$**  fields of the **Points** box. ( $z0$  will appear only for 3D problems.)

### 3.6.6. Point Properties Specific to Line Fiber Injections

In a line fiber injection the starting points of the fibers are arranged on a line at a constant distance (see [Figure 3.5: Line Injections \(p. 172\)](#)). You have to specify the coordinates of the starting point and the end point of this line. Click the **Injection Point Properties** tab in the **Set Fiber Injection Properties** dialog box ([Figure 3.3: The Set Fiber Injection Properties Dialog Box \(p. 169\)](#)). In the **Points** region, set the  $x$ ,  $y$ , and  $z$  coordinates in the  **$x0$ ,  $y0$ , and  $z0$**  fields for the starting point of the line and the  $x$ ,  $y$ , and  $z$  coordinates in the  **$x1$ ,  $y1$ , and  $z1$**  fields for the end point of the line. ( $z0$  and  $z1$  will appear only for 3D problems.)

**Figure 3.5: Line Injections**



In addition to the coordinates, you have to set the number of fibers defined in the line injection by entering the appropriate value in the **Point Density Edge1** field. See [Figure 3.5: Line Injections \(p. 172\)](#) for an example of a line fiber injection with a **Point Density Edge1** of 5.

### 3.6.7. Point Properties Specific to Matrix Fiber Injections

In a matrix fiber injection the starting points of the fibers are arranged in several rows having the shape of a rectangle or a parallelogram. Each row has the same distance to the previous and is divided into equal sections (see [Figure 3.6: Matrix Injections \(p. 173\)](#)).

You have to specify the coordinates of the starting point and the end of point of the first row and the coordinates where the last row should start. Click the **Injection Point Properties** tab in the **Set Fiber Injection Properties** dialog box ([Figure 3.3: The Set Fiber Injection Properties Dialog Box \(p. 169\)](#)). In the **Points** region, set the  $x$ ,  $y$ , and  $z$  coordinates in the  **$x0$ ,  $y0$ , and  $z0$**  fields for the starting point and the  $x$ ,  $y$ , and  $z$  coordinates in the  **$x1$ ,  $y1$ , and  $z1$**  fields for the end point of the first row of fibers. The  $x$ ,  $y$ , and  $z$  coordinates of the starting point of the last row have to be entered in the  **$x2$ ,  $y2$ , and  $z2$**  fields.

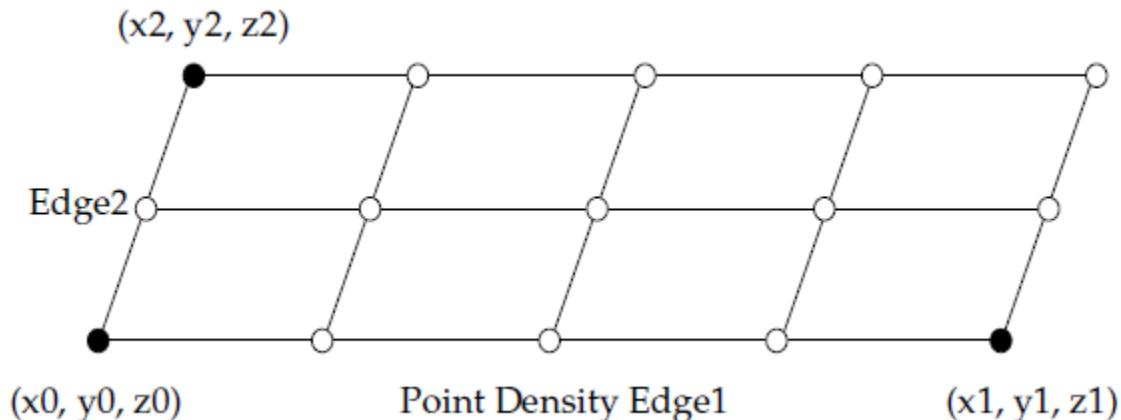
At each row, the number of fibers specified in the **Point Density Edge1** field are injected. The number of rows to be injected is specified in **Edge2**.

You can double check the number of fibers computed in this fiber injection if you inspect the **Numbers of Fibers Computed** field.

### Important

Note that this fiber injection type is only available for 3D problems.

**Figure 3.6: Matrix Injections**



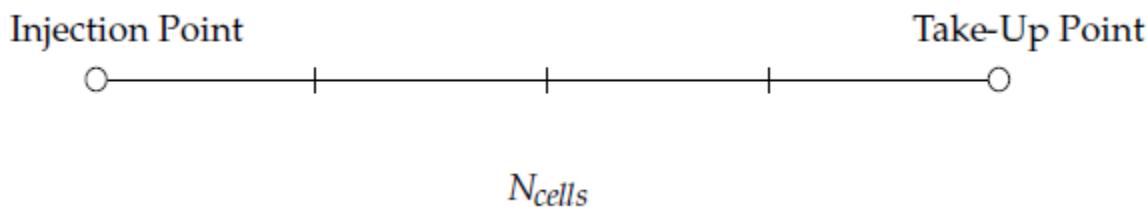
### 3.6.8. Define Fiber Grids

Each fiber consists of a straight line between the injection point and the take-up point. It is divided into a number of finite volume cells. Every fiber defined in a fiber injection has its own grid, which you can specify if you click the **Grid Properties** tab in the **Set Fiber Injection Properties** dialog box.

#### 3.6.8.1. Equidistant Fiber Grids

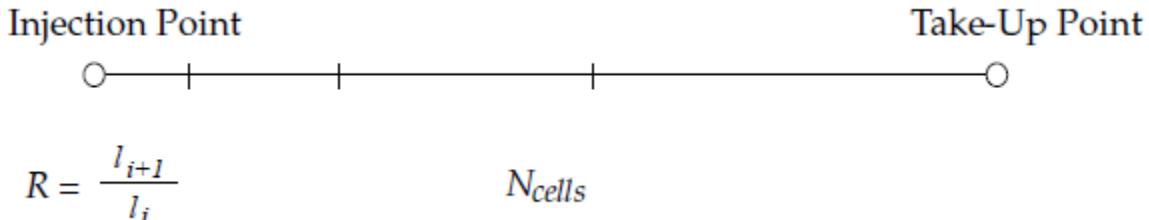
To define an equidistant grid for the fibers select **equidistant** from the **Grid Type** drop-down list. Specify the **Number of Cells** into which every fiber of the fiber injection will be divided.

**Figure 3.7: Equidistant Fiber Grid**



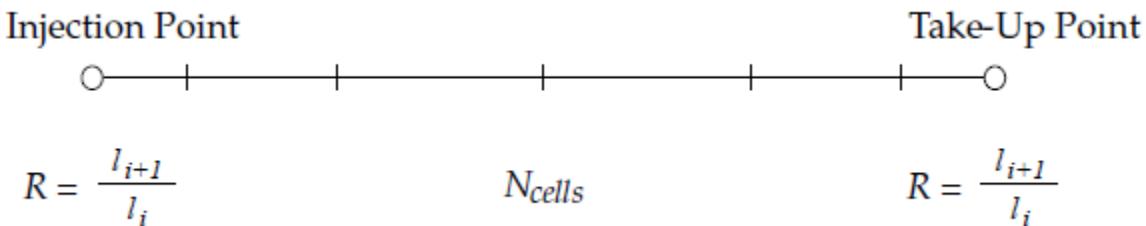
#### 3.6.8.2. One-Sided Fiber Grids

A one-sided grid is graded near the injection point of the fiber. To define a one-sided grid for the fibers select **one-sided** from the **Grid Type** drop-down list. Specify the **Number of Cells** into which every fiber of the fiber injection will be divided. In addition, you have to specify the ratio,  $R$ , between each subsequent fiber cell in the **Grid Growth Factor at Injection Point** field. Values larger than 1 refine the grid, while values smaller than 1 coarsen the grid.

**Figure 3.8: One-Sided Fiber Grid**

### **3.6.8.3. Two-Sided Fiber Grids**

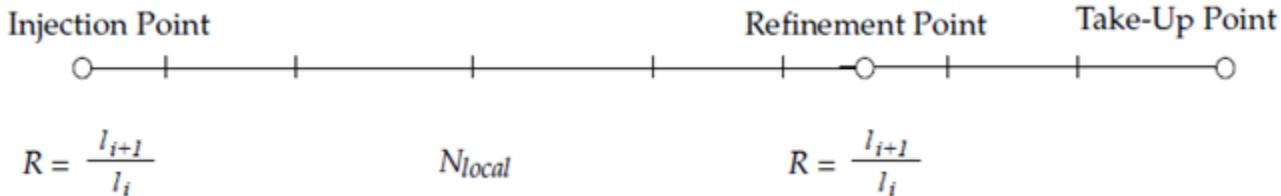
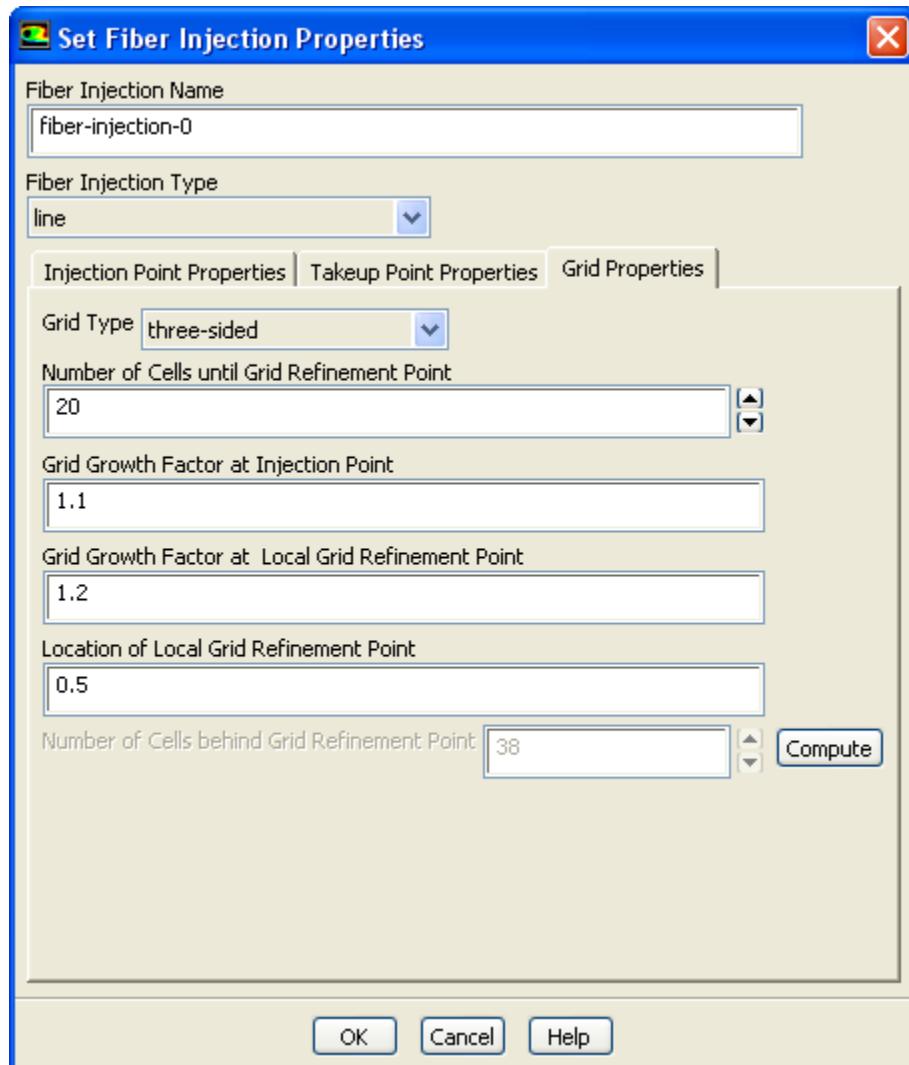
A two-sided grid is graded near the injection point as well as at the take-up point of the fiber. To define a two-sided grid for the fibers, select **two-sided** from the **Grid Type** drop-down list. Specify the **Number of Cells** into which every fiber of the fiber injection will be divided. You also have to specify the ratio,  $R$ , between each subsequent fiber cell at the injection point in the **Grid Growth Factor at Injection Point** field and the ratio,  $R$ , near the take-up point in the **Grid Growth Factor at Takeup Point** field. Values larger than 1 refine the grid, while values smaller than 1 will coarsen it.

**Figure 3.9: Two-Sided Fiber Grid**

### **3.6.8.4. Three-Sided Fiber Grids**

A three-sided grid consists of three sides where the fiber grid can be graded. The first side is near the injection point. The other two sides are around a refinement point within the fiber grid. Both sides at the refinement point are graded at the same ratio between the fiber grid cell lengths. To define a three-sided grid for the fibers, select **three-sided** from the **Grid Type** drop-down list (see [Figure 3.11: Defining a Three-Sided Fiber Grid Using the Set Fiber Injection Properties Dialog Box \(p. 175\)](#)). Specify the **Number of Cells until Grid Refinement Point**, the **Grid Growth Factor at Injection Point**, and the **Grid Growth Factor at Local Grid Refinement Point**. You also have to specify the location of the grid refinement point in the **Location of Local Grid Refinement Point** field in a dimensionless way. The value you have to enter is relative to the fiber length and may be between 0 and 1. Values larger than 1 refine the grid, while values smaller than 1 coarsen the grid.

Click the **Compute** button to estimate the **Number of Cells behind Grid Refinement Point**.

**Figure 3.10: Three-Sided Fiber Grid****Figure 3.11: Defining a Three-Sided Fiber Grid Using the Set Fiber Injection Properties Dialog Box**

### 3.7. User-Defined Functions (UDFs) for the Continuous Fiber Model

The continuous fiber model enables you to apply custom correlations for drag, heat transfer, mass transfer coefficients, as well as material properties to the fibers by means of user-defined functions (UDFs). You are provided with a C template file named `fiber_fluent_interface.c` that contains source code for the drag, heat transfer, mass transfer coefficient, and the fiber property UDFs. You will need to modify this UDF source file to suit your application, compile it, and hook the resulting UDF object(s) to the fiber model. This process is described below.

For more information, see the following sections:

[3.7.1. UDF Setup](#)[3.7.2. Customizing the fiber\\_fluent\\_interface.c File for Your Fiber Model Application](#)[3.7.3. Compile Fiber Model UDFs](#)[3.7.4. Hook UDFs to the Continuous Fiber Model](#)

## **3.7.1. UDF Setup**

Before you can edit the UDF template and create your custom UDF for drag, heat transfer or mass transfer coefficient, you must copy the `fiber` directory to your working directory. The `fiber` directory contains all of the libraries and support files that are required to compile UDFs for the fiber model and build shared libraries for the architecture that you specify.

### **3.7.1.1. Linux Systems**

Make a local copy of the `fiber` directory by copying it from the path below to your working directory.

`path/ansys_inc/v170/fluent/fluent17.0.0/addons/fiber/`

where `path` is the directory in which you have installed ANSYS Fluent.

### **3.7.1.2. Windows Systems**

1. Open a Visual Studio .NET DOS prompt.
2. Make a local copy of the `fiber` folder (not a symbolic link) by copying it from the folder below to your working folder.

`path\ANSYS_Inc\v170\fluent\fluent17.0.0\addons\fiber\`

where `path` is the folder in which you have installed ANSYS Fluent (by default, the path is `C:\Program Files`).

## **3.7.2. Customizing the fiber\_fluent\_interface.c File for Your Fiber Model Application**

Now that you have copied the `fiber` directory to your working directory, you can edit the UDF template file and customize it to fit your model needs.

1. In your working directory, change directories to `fiber/src`. The `/src` directory contains the UDF template source file `fiber_fluent_interface.c`.
2. In the `/src` directory, use any text editor and edit `fiber_fluent_interface.c`.
3. Scroll down to the bottom of the `fiber_fluent_interface.c` file to the section that contains concatenated functions for friction factor (drag coefficient), heat transfer coefficient, mass transfer coefficient, fiber viscosity, fiber density, fiber thermal conductivity, fiber specific heat, and vapor liquid equilibrium, respectively. These are the UDFs that you can modify and customize.

- 
4. Edit the function(s) you require for your particular application. Save `fiber_fluent_interface.c` and overwrite the existing file.
- 

### **Important**

Do not save the file with another name because it will not be recognized by the system.

---

### **Important**

The function names of the templates `user_friction_factor`, `user_heat_transfer_coefficient`, and `user_mass_transfer_coefficient` *must not* be altered because they are called by other routines in the continuous fiber model.

---

### **3.7.2.1. Example: Heat Transfer Coefficient UDF**

Below is an example of a heat transfer coefficient UDF that is defined in `fiber_fluent_interface.c`. The function is taken from Kase and Matsuo [3] (p. 191) and is implemented as the `kase-matsuo-1` option in the fiber model. The function name `user_heat_transfer_coefficient` *must not*, under any circumstances, be altered because it is called by other functions in the continuous fiber model.

There are two arguments to the `user_heat_transfer_coefficient` UDF: `Fiber` and `Local_Fiber_Data_Type`. `Fiber *f` is a pointer to the fiber structure that contains information about the fiber and `Local_Fiber_Data_Type *fd` accesses temporary variables that are needed during the calculation of the fiber.

In the sample UDF below, a loop is performed over all fiber grid cells using the macro `FIB_N(f)` which represents the number of grid cells for the fiber `f`. The Reynolds number is computed based on the relative velocity,  $u_f - u_{par}$  ( $\text{ABS}(\text{FIB\_C\_U}(f, i) - \text{fd}->\text{up}[i])$ ), the fiber diameter; `FIB_C_D(f, i)`, the density of the surrounding fluid; `fd->rho[i]`, and the viscosity of the surrounding fluid; `fd->vis[i]`. The heat transfer coefficient  $\alpha$  is computed from the Nusselt number using the thermal conductivity of the surrounding fluid,  $\lambda$ , (`fd->k[i]`) and is stored in `fd->alpha[i]`.

```
void
user_heat_transfer_coefficient(Fiber *f, Local_Fiber_Data_Type *fd)
{
    int i;
    real Red, Nud;

    /* model from Kase/Matsuo (1967) */
    for (i=0; i<FIB_N(f); i++)
    {
        /* compute Reynolds number based on relative velocity */
        Red = ABS(FIB_C_U(f,i)-fd->up[i])*FIB_C_D(f,i)*fd->rho[i]/fd->vis[i];
        Nud = 0.42*pow(Red, 0.334);
        /* store heat transfer coefficient for latter use */
        fd->alpha[i] = Nud*fd->k[i]/FIB_C_D(f,i);
    }
}
```

All variables and macros that are used in `user_heat_transfer_coefficient` are defined in header files provided with the continuous fiber model. For example, you will find the type definition `Fiber` and the macros that are used to access variables of a single fiber in the header file `fiber.h`.

Temporary variables used in the type definition of Local\_Fiber\_Data\_Type can be found in the header file fib-mem.h.

---

**Important**

Note that you must not modify the header files provided with the continuous fiber model. Otherwise the compiled library will not be compatible with ANSYS Fluent and will show run time errors.

---

### 3.7.2.2. Example: Fiber Specific Heat Capacity UDF

You can also calculate fiber material properties using UDFs. ANSYS Fluent provides a UDF template for every fiber property. You can modify the UDF templates for your specific case.

The template for the fiber density is as follows:

```
real User_Fiber_Specific_Heat(Fiber *f, real temp, real liq, int i)
{
    real yi[MAX_SPE_EQNS];
    real cp_p;
    /* compute cp based on FLUENT's standard procedure selected in materials panel */
    cp_p = Specific_Heat(FIB_MATERIAL_POLYMER(f), temp, 0., yi);
    if (FIB_DRY_SPINNING(f))
    {
        real cp_s;
        cp_s = Specific_Heat(FIB_MATERIAL_SOLVENT(f), temp, 0., yi);
        return cp_p*(1.-liq)+cp_s*liq;
    }
    return cp_p;
}
```

---

**Important**

The function names of the templates User\_Friction\_Factor, User\_Heat\_Transfer\_Coefficient, User\_Mass\_Transfer\_Coefficient, User\_Fiber\_Viscosity, User\_Fiber\_Density, User\_Fiber\_Thermal\_Conductivity, User\_Fiber\_Specific\_Heat, and User\_Fiber\_Vle must not be altered since they are called by other routines of the continuous fiber model.

---

All variables and macros defined in the header files are provided with the continuous fiber model. For example, you will find the type definition Fiber and the macros to access variables of a single fiber in the header file fiber.h. The temporary variables used in the type definition of Local\_Fiber\_Data\_Type can be found in the header file fib-mem.h.

---

**Important**

You must not modify the header files provided with the continuous fiber model. Otherwise the compiled library will not be compatible with ANSYS Fluent resulting in runtime errors (messages will be printed in the console).

---

### 3.7.3. Compile Fiber Model UDFs

Once you have customized the UDF template fiber\_fluent\_interface.c for your fiber model application, you are now ready to compile the source file using the text interface. Follow the procedure below for Linux Systems and Windows Systems, respectively.

### 3.7.3.1. Linux Systems

In the `/src` directory that is under `/fiber` in your working directory, run the `Makefile` command that will compile your fiber mode UDF and build a shared library for the architecture you are running. To do this, type the following command at the prompt:

```
make -f Makefile-client FLUENT_ARCH=your_arch
```

where `your_arch` is replaced by the architecture of the machine you are running (for example, `lnx86`).

For example, if your computer architecture is `lnx86` type the following command in a terminal session:

```
make -f Makefile-client FLUENT_ARCH=lnx86
```

To identify the architecture of the machine you are running on, scroll up the ANSYS Fluent console window to the message that begins with "Starting".

When you run the `makefile` process, the source code (`fluent_fiber_interface.c`) will be compiled into object code and a shared library will be built for the computer architecture and version of ANSYS Fluent you are running. Messages about the compile/build process will be displayed on the console window. You can view the compilation history in the log file that is saved in your working directory. Below is an example of console messages for a `lnx86` architecture running a 2D version of ANSYS Fluent.

```
Working...
for d in lnx86[23]*; do \
( \
cd $d; \
for f in ../../src*.ch ../../src/makefile; do \
if [ ! -f 'basename $f' ]; then \
echo "# linking to " $f " in" $d; \
ln -s $f .; \
fi; \
done; \
echo ""; \
echo "# building library in" $d; \
make -kmakelog 2&1; \
cat makelog; \
) \
done
# linking to ... myudf.c in lnx86/2d

# building library in lnx86/2d
make[1]: Entering directory ..../udf_names.c
# Generating udf_names
make[2]: Entering directory ..../fluent_fiber_interface.c
make libudf.so ...
# Compiling udf_names.o ...
# Compiling profile.o ...
# Linking libudf.so ...
make[2]: Leaving directory ..../udf_names.c
make[1]: Leaving directory ..../fluent_fiber_interface.c

You can also see the 'log'-file in
the working directory for compilation history

Done.
```

### 3.7.3.2. NT/Windows Systems

In the `/src` directory that is under `/fiber` in your working directory, run the `Makefile` command that will compile your fiber mode UDF and build a shared library for it for the architecture you are running. To do this, type the following command at the prompt:

```
nmake /f makefile_master-client.nt
```

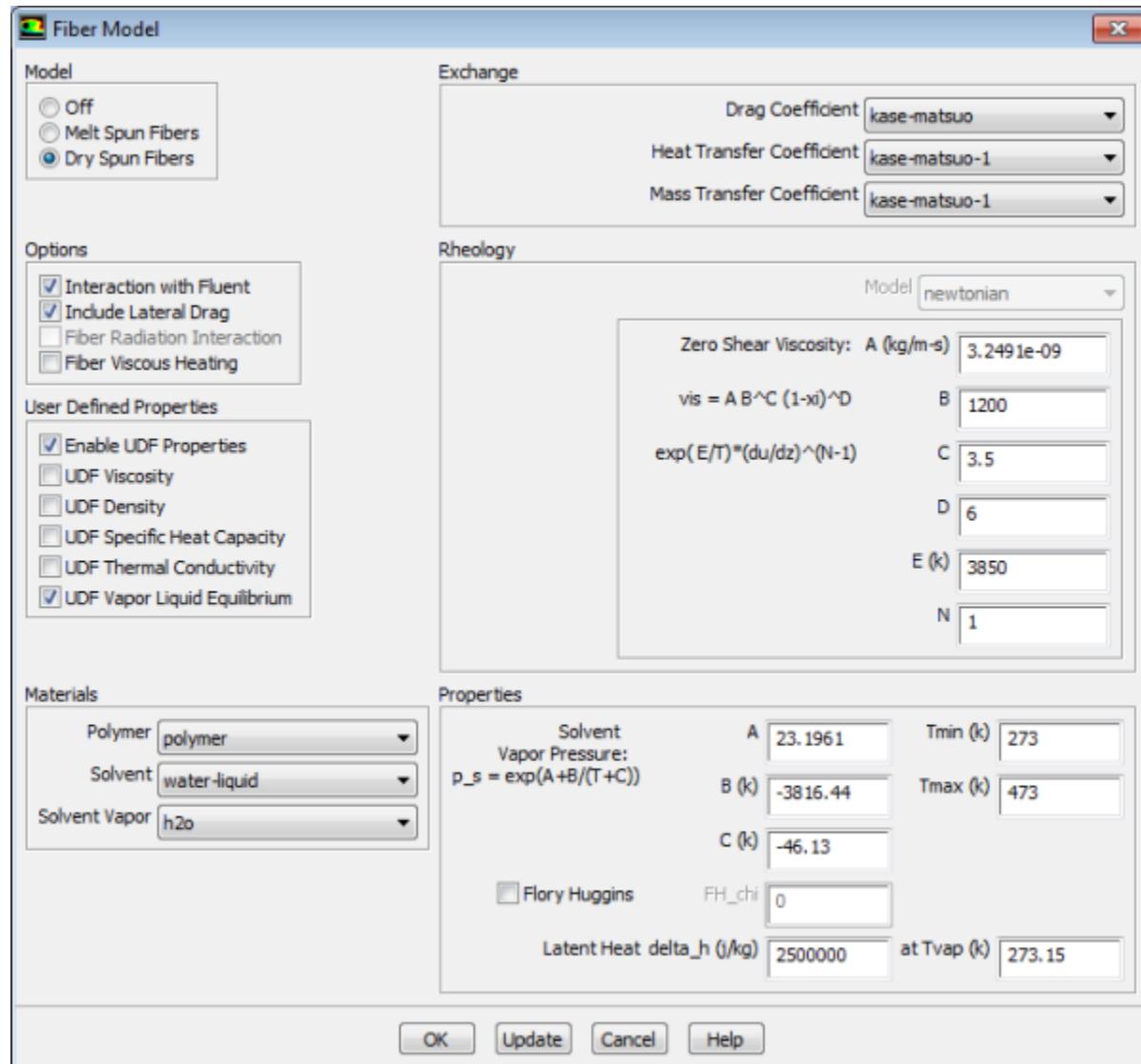
To identify the architecture of the machine you are running on, scroll up the ANSYS Fluent console window to the message that begins with "Starting".

When you run the `makefile` process, the source code (`fluent_fiber_interface.c`) will be compiled into object code and a shared library will be built for the computer architecture and version of ANSYS Fluent you are running. Messages about the compile/build process will be displayed on the console window. You can view the compilation history in the log file that is saved in your working directory.

### 3.7.4. Hook UDFs to the Continuous Fiber Model

Once you have successfully compiled your continuous fiber model UDF(s), the **user-defined** option will appear in drop-down lists for parameters under **Exchange** in the **Fiber Model** dialog box ([Figure 3.12: The Fiber Model Dialog Box \(p. 181\)](#)). To hook your UDF to a particular fiber model parameter, simply select **user-defined** from the drop-down list for **Drag Coefficient**, **Heat Transfer Coefficient**, or **Mass Transfer Coefficient** and click **OK**.

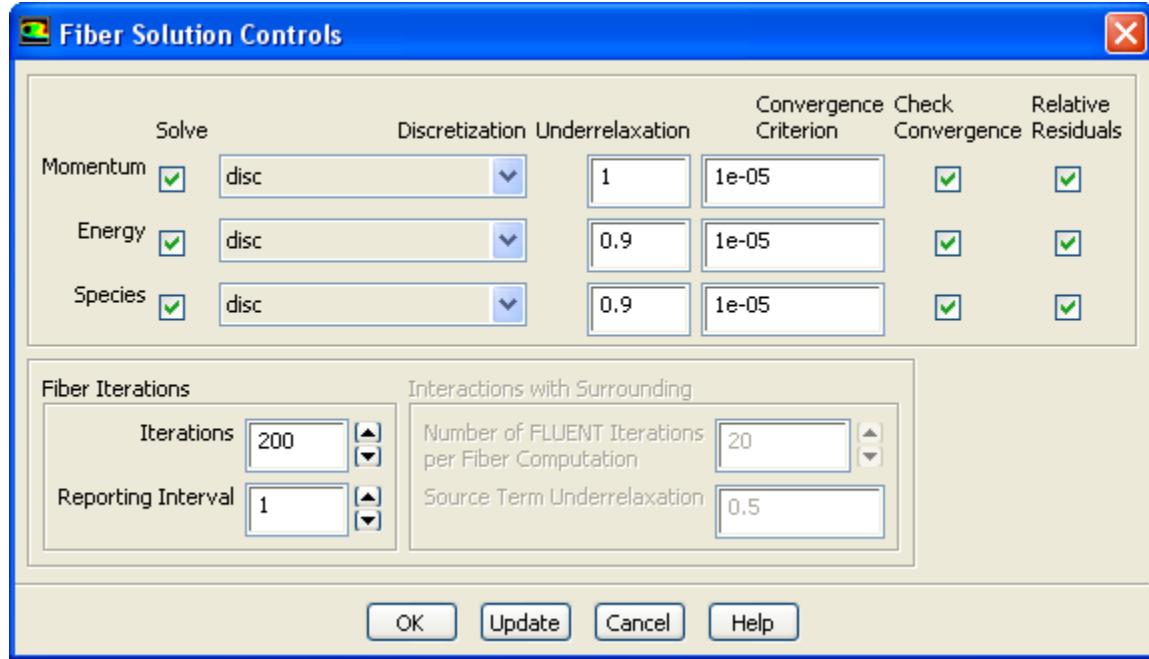
If you want to use UDFs to define fiber material properties, select the **Enable UDF Properties** check box, and then enable individual property UDFs under the **User Defined Properties** group box.

**Figure 3.12: The Fiber Model Dialog Box**

## 3.8. Fiber Model Solution Controls

To access the **Fiber Solution Controls** dialog box, go to

**Setup** → **Models** → **Fiber-Controls** **Edit...**

**Figure 3.13: Fiber Solution Controls Dialog Box**

The **Fiber Solution Controls** dialog box enables you to set common solution parameters for the fiber equations and their coupling with the surrounding fluid.

### Solve

is used to enable and disable the solution of the fiber equations for **Momentum** ([Equation 2.4 \(p. 140\)](#)), **Energy** ([Equation 2.11 \(p. 141\)](#)), and **Species** ([Equation 2.1 \(p. 139\)](#)). When switching off the solution of one of these equations, it is not computed for all fibers defined in your model.

### Discretization

provides a drop-down list where you can assign to each of the fiber equations one of three different discretization schemes explained in [Discretization of the Fiber Equations \(p. 142\)](#).

### Underrelaxation

contains all under-relaxation factors for all fiber equations that are being solved in the continuous fiber model. See [Under-Relaxation \(p. 142\)](#) for additional background information and see [Solution Strategies \(p. 152\)](#) for how to make use of under-relaxation factors in your solution strategy.

### Convergence Criterion

is used to stop the fiber iterations when the residuals of all fiber equations are below the prescribed criteria. You can define a separate convergence criterion for every fiber equation.

### Check Convergence

must be turned on if you want to compare the residuals of the fiber equations with the **Convergence Criterion**. If you turn this option off, the given number of fiber iterations will be computed.

### Relative Residuals

are used to compute the change of the residual of two subsequent iterations relative to the residual of the last iteration by applying [Equation 2.20 \(p. 144\)](#). The result of this is compared to the **Convergence Criterion** to check whether convergence has been achieved.

### Iterations

defines the number of **Fiber Iterations** performed every time the fiber equations are updated.

**Reporting Interval**

sets the number of fiber iterations that will pass before the residuals will be printed.

You can reduce the output to the last fiber iteration by specifying **Reporting Interval** as 0. This is recommended when performing a solution that is coupled with the surrounding flow.

**Number of Fluent Iterations per Fiber Computation**

sets the number of Fluent iterations before the fiber equations are updated in a solution that is coupled with the surrounding flow.

**Source Term Underrelaxation**

factor is used to under-relax the fiber source term exchange to the surrounding fluid. In a converged solution this value does not influence your predictions.

For additional information on how to set and choose values for the options in the **Fiber Solutions Control** dialog box, see [Solution Strategies \(p. 152\)](#).

## 3.9. Postprocessing for the Continuous Fibers

After you have completed your inputs and performed any coupled calculations, you can display the location of the fibers and source terms of the fibers, and you can write fiber data to files for further analysis of fiber variables.

The following data can be displayed using graphical and alphanumeric reporting facilities:

- Graphical display of fiber locations
- Exchange terms with surrounding fluid
- Fiber solution variables.

For more information, see the following sections:

[3.9.1. Display of Fiber Locations and Grid Points](#)

[3.9.2. Exchange Terms of Fibers](#)

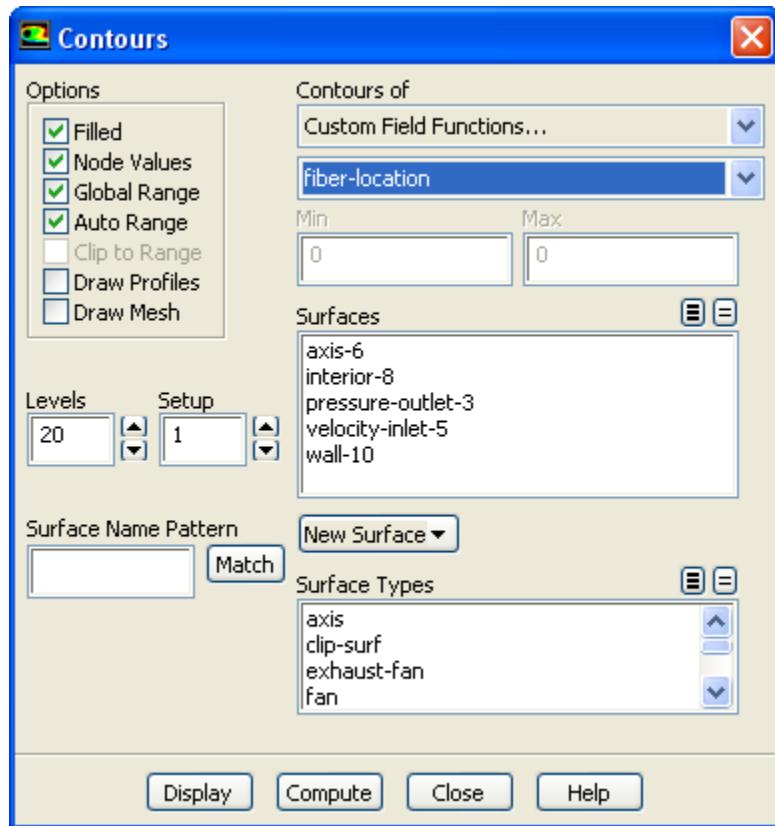
[3.9.3. Analyzing Fiber Variables](#)

[3.9.4. Running the Fiber Module in Parallel](#)

### 3.9.1. Display of Fiber Locations and Grid Points

When you have defined fiber injections, as described in [Defining Fibers \(p. 163\)](#), you can display the location of the fibers using ANSYS Fluent's **Contour** plot facility ([Figure 3.14: Displaying Fiber Locations Using the Contours Dialog Box \(p. 184\)](#)).

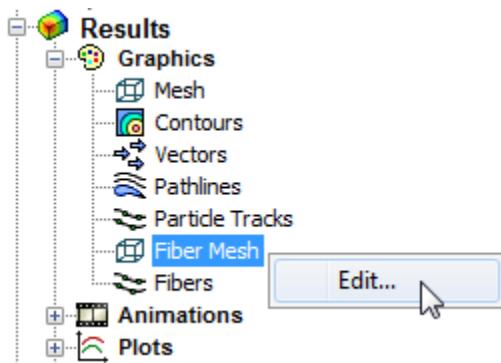
 Results → Graphics → Contours  Edit...

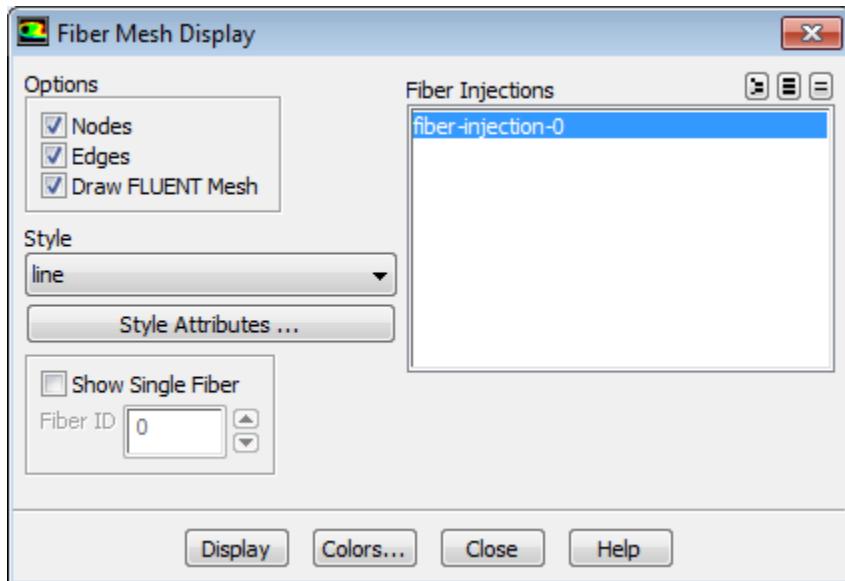
**Figure 3.14: Displaying Fiber Locations Using the Contours Dialog Box**

In the **Contours of** drop-down list, select **Custom Field Functions...**, then select **fiber-location**. The values in this field are between zero and the number of fiber cells in an ANSYS Fluent grid cell.

You may generate iso-surfaces of constant values of **fiber-location** to display the fibers in 3D problems.

You can also display the locations of the fibers and their grid discretization using the **Fiber Mesh Display** dialog box, which can be accessed from the **Results/Graphics/Fiber Mesh** tree item.

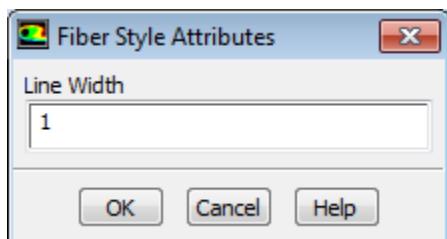


**Figure 3.15: The Fiber Mesh Display Dialog Box**

The following mesh discretization options are available:

- **Nodes:** displays the end points of fiber volume cells
- **Edges:** displays a line along the path of fibers in the domain
- **Draw Fluent mesh:** Enables the display of the CFD mesh geometry with the mesh display settings specified in the **Mesh Display** dialog box. The **Mesh Display** dialog box will appear automatically when you enable the **Draw Fluent mesh** option.

You can set the fiber **Line Width** in the **Fiber Style Attribute** dialog box accessed by clicking the **Style Attributes...** button.

**Figure 3.16: The Fiber Style Attributes Dialog Box**

For the injections selected in the **Fiber Injections** selection list, you can display either all injection fibers or single fibers with a specified **Fiber ID** (when the **Show Single Fiber** option is selected).

### 3.9.2. Exchange Terms of Fibers

The continuous fiber model computes and stores the exchange of momentum, heat, mass, and radiation in each control volume in your ANSYS Fluent model. You can display these variables graphically by drawing contours, profiles, and so on. They all are contained in the **Custom Field Functions...** category of the variable selection drop-down list that appears in postprocessing dialog boxes:

#### **fiber-mass-source**

defined by [Equation 2.22 \(p. 145\)](#).

**fiber-x-momentum-source**

is the x-component of the momentum exchange, defined by [Equation 2.21 \(p. 144\)](#).

**fiber-y-momentum-source**

is the y-component of the momentum exchange, defined by [Equation 2.21 \(p. 144\)](#).

**fiber-z-momentum-source**

is the z-component of the momentum exchange, defined by [Equation 2.21 \(p. 144\)](#).

**fiber-energy-source**

defined by [Equation 2.23 \(p. 145\)](#).

**fiber-dom-absorption**

defined by [Equation 2.24 \(p. 146\)](#).

**fiber-dom-emission**

defined by [Equation 2.25 \(p. 146\)](#).

Note that these exchange terms are updated and displayed only when coupled computations are performed.

### 3.9.3. Analyzing Fiber Variables

You can use ANSYS Fluent's **Plot** facilities to analyze fiber solution variables such as fiber velocity, temperature, diameter, and so on.

#### 3.9.3.1. XY Plots

To investigate fiber variables you have to generate an xy-file for every variable using the following procedure from the text command interface (assuming that you have already defined a fiber injection):

1. Store an xy-file with the variable of interest.

```
continuous-fiber → print-xy
```

2. Enter the fiber injection of interest when asked for

```
Fiber Injection name>
```

3. Specify a variable for x-column. This will be used as abscissa in the file plot.

4. Specify another variable for y-column, which will be used as ordinate in the file plot.

5. Specify a file name when asked for XY plot file name.

6. Load the file with ANSYS Fluent's xy plot facilities, described in the ANSYS Fluent manual.

The following variables can be entered as x-column and y-column:

- axial drag
- conductivity
- curvature
- diameter

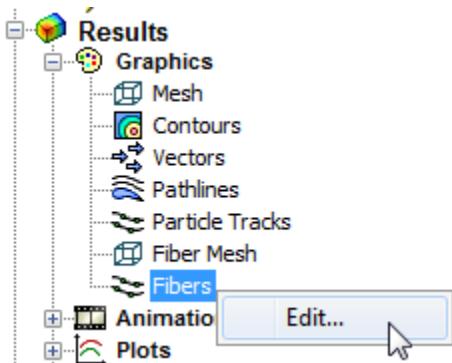
- enthalpy
- evaporated mass flux
- heat transfer coefficient
- lateral drag
- length
- mass flow
- mass fraction
- mass transfer coefficient
- Nusselt number
- Sherwood number
- Reynolds number
- specific heat
- surface mass fraction
- temperature
- tensile force
- user variable
- velocity
- velocity gradient
- viscosity

If a fiber injection consists of several fibers, the data of all fibers in this fiber injection will be stored in the xy-file.

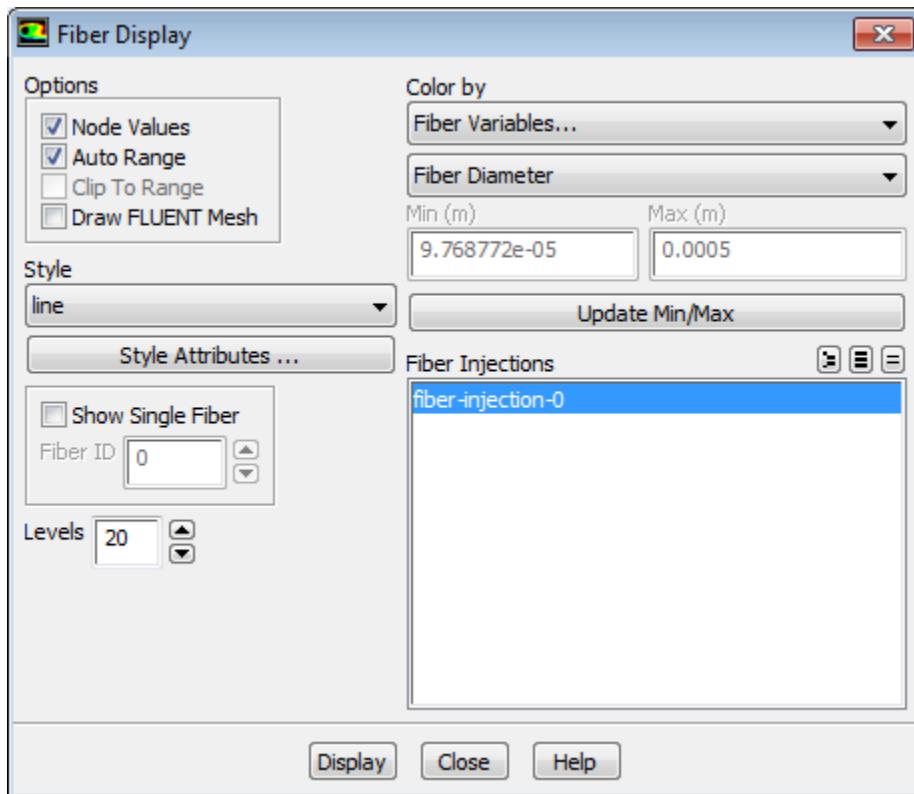
In addition to this, you can store a fiber history file for a fiber injection as described in [Print Fiber Injections \(p. 166\)](#). This can be used to analyze a single fiber using ANSYS Fluent xy-plot facilities or by external postprocessing programs.

### 3.9.3.2. Fiber Display

To visualize injection fibers, you can use the **Fiber Display** dialog box, which can be accessed from the **Results/Graphics/Fibers** tree item.



**Figure 3.17: The Fiber Display Dialog Box**



The **Fiber Display** dialog box offers the same display options as those in the **Fiber Mesh Display** dialog box (Figure 3.15: The Fiber Mesh Display Dialog Box (p. 185)). Similarly, for the injections selected in the **Fiber Injections** selection list, you can display either all injection fibers or single fibers with a specified **Fiber ID** (when the **Show Single Fiber** option is selected).

You can color the fibers by the fiber variables listed in XY Plots (p. 186) or by the following additional fiber variables:

- **Fiber ID**
- **Fiber Density**
- **Fiber Tension**

You can control the smooth gradation of fiber colors by specifying the number of fiber **Levels**.

### 3.9.4. Running the Fiber Module in Parallel

The Fiber Module can be used in serial as well as in parallel calculations. Simulations in parallel require no additional input. The fiber iterations are performed entirely on the host process, whereas the flow field is parallelized. As a result, the overall performance may be limited by the host process if the fiber calculations are more time-consuming than the parallel flow-field iterations. Therefore, when simulating a large number of fibers, it is recommended to spawn the host process on a suitable machine. Refer to [Parallel Processing in the \*Fluent User's Guide\*](#) for further information on how to set up parallel calculations.



---

# Bibliography

- [1] P. J. Flory. "Thermodynamics of High Polymer Solutions". *Journal of Chemical Physics*. 10. 51–61. 1942.
- [2] B. Gampert. *Grenzschichttheoretische Probleme des aerodynamischen Schmelzspinnprozesses*. PhD Thesis from the Technical University of Berlin. Berlin, Germany 1973.
- [3] S. Kase and T. Matsuo. "Studies on Melt Spinning. II. Steady-State and Transient Solution of Fundamental Equations Compared with Experimental Results". *Journal of Applied Polymer Science*. 11. 251–287. 1967.
- [4] Y. Ohzawa, Y. Nagano, and T. Matsuo. "Fundamental Equations of Dry Spinning with an Example Calculation". *Proceedings, 5th Intern. Congress on Rheology*. 393–408. Kyoto. 1968.
- [5] S. V. Patankar. *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill. Washington, New York, London 1980.
- [6] C. H. Rexroth, H. J. Bauer, and S. Wittig. "DISC—An efficient method for the discretization of convection on unstructured grids". *Aerospace Science and Technology*. 6. 1997.
- [7] H. Schlichting and K. Gersten. *Grenzschicht-Theorie*. Springer. Berlin, Heidelberg, New York, London 1997.



---

---

## **Part IV: ANSYS Fluent Macroscopic Particle Module**

[Using This Manual \(p. cxcv\)](#)

- [1. Introduction \(p. 197\)](#)
- [2. Macroscopic Particle Model Theory Module \(p. 199\)](#)
- [3. Using the Macroscopic Particle Model \(p. 205\)](#)

[Bibliography \(p. 225\)](#)

---

---

---

# Using This Manual

---

This preface is divided into the following sections:

**1. The Contents of This Manual**

The ANSYS Fluent Macroscopic Particle Module Manual provides information about using the Macroscopic Particle Model available in ANSYS Fluent. In this manual, you will find a theoretical background for the macroscopic particle model used in ANSYS Fluent, and a description of how to use the model in your CFD simulations.

This part is divided into the following chapters:

- [Introduction \(p. 197\)](#)
- [Macroscopic Particle Model Theory \(p. 199\)](#)
- [Using the Macroscopic Particle Model \(p. 205\)](#)
- [Bibliography \(p. 225\)](#)



---

---

## Chapter 1:Introduction

---

Traditional Lagrangian discrete phase models only apply when particle sizes are small enough to be regarded as point masses within a single cell and when the total particle volume is insignificant within the flow domain volume. In these applications particle-particle and particle-flow interactions are evaluated in terms of impulse, heat, and mass transfer.

However, the total particle volume must be considered when a particle size is larger than several fluid cells because it will affect all the cells within the flow domain volume. ANSYS Fluent Macroscopic Particle Model (MPM) predicts the behavior of large (macroscopic) particles and their interaction with the fluid flow, walls, and with other particles.



---

## Chapter 2: Macroscopic Particle Model Theory

---

The Macroscopic Particle Model (MPM) is a UDF-based quasi-direct numerical approach for tracking macroscopic particles [4] (p. 225). The MPM is applicable to Lagrangian particulate flows that cannot be solved using conventional point-mass particle models. In such flows, the particle size cannot be neglected. In these situations, particle volume must be considered when modeling hydrodynamics and wall effects.

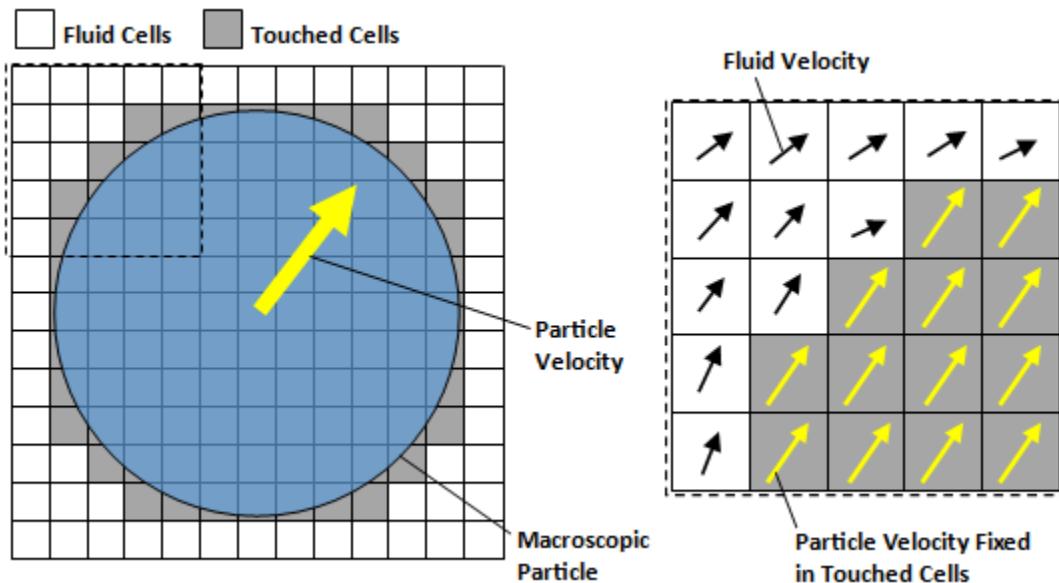
The MPM model provides a special treatment that accounts for the following:

- Flow blockage and momentum exchange
- Calculation of drag and torque on particles
- Particle-particle and particle-wall collision, and friction dynamics
- Particle deposition and buildup
- Particle-particle and particle-wall attraction forces

In the MPM approach [1] (p. 225), each macroscopic particle spans several computational cells. Each particle is represented by a sphere with six degrees of freedom to account for the particle translational and rotational motion. The particle is injected in the flow domain at the beginning of a flow time step. The particle is assumed to be touching a computational cell during the time step if one or more nodes of the cell are located inside the particle volume. Each particle transport equation is solved in a Lagrangian reference frame.

### 2.1. Momentum Transfer to Fluid Flow

At each time step, a volume-fraction weighted velocity between the particle velocity (translational and rotational) and the flow velocity in the cell from the last time step is assigned to the fluid cells touched by the particle. Additionally, the cell velocity of all cells touched by the particle is forced towards particle velocity extrapolated to the cell center using source terms. As a consequence, the flow velocity of touched cells is dominated by the particle velocity

**Figure 2.1: Fixing Velocities in Fluid Cells Touched by the Particle**

## 2.2. Fluid Forces and Torques on Particle

At each time step, drag and torque acting on the particle are computed using explicit expressions involving the velocity, pressure, and shear stress distribution in the fluid cells surrounding the particle [2] (p. 225). The total fluid forces and torques acting on a macroscopic particle in the direction  $\mathbf{i}$  consist of virtual mass, pressure, and viscous fluid components:

$$R_i = R_{m,i} + R_{p,i} + R_{v,i} \quad (2.1)$$

The  $i^{\text{th}}$  virtual mass component of the fluid force and torque experienced by a particle,  $R_{m,i}$ , is calculated as the integral of the rate of change of momentum for all fluid cells within a particle volume:

$$R_{m,i} = \left( \sum_{\substack{\text{volume} \\ \text{cells}}} m_f V_{f,i} - \sum_{\substack{\text{volume} \\ \text{cells}}} m_f V_{p,i} \right) \frac{1}{\Delta t} \quad (2.2)$$

where  $m_f$  is the cell fluid mass;  $V_{f,i}$  and  $V_{p,i}$  are the fluid and particle velocities in the direction  $i$ , respectively; and  $\Delta t$  is the flow time step.

The  $i^{\text{th}}$  pressure component of the fluid force and torque acting on the particle surface,  $R_{p,i}$ , is calculated based on the pressure distribution around the particle:

$$R_{p,i} = \sum_{\substack{\text{surface} \\ \text{cells}}} P d^2 \frac{r_i}{|\vec{r}|} \quad (2.3)$$

where  $P$  is the pressure,  $d^2$  is the approximated area of a particle surface in a fluid cell touching the particle,  $\vec{r}$  is the radius vector from the fluid cell center to the particle center, and  $r_i$  is the Cartesian component of vector  $\vec{r}$  in the  $i^{\text{th}}$  direction.

The  $i^{\text{th}}$  viscous component of the fluid force and torque acting on a particle surface,  $R_{v,i}$ , is calculated based on the shear stress distribution around the particle:

$$R_{v,i} = \sum_{\substack{\text{surface} \\ \text{cells}}} \sum_j \vec{\tau}_{ij} d^2 \left( -\frac{r_j}{|\vec{r}|} \right) \quad (2.4)$$

where  $\vec{\tau}_{ji}$  is the shear stress in the positive  $i^{\text{th}}$  direction acting on a plane perpendicular to the  $j^{\text{th}}$  direction, and  $r_j$  is the Cartesian component of vector  $\vec{r}$  in the  $j^{\text{th}}$  direction.

Based on fluid forces and torques, the new particle position, velocities, and accelerations are calculated at each flow time step.

## 2.3. Particle/Particle and Particle/Wall Collisions

ANSYS Fluent provides a hard sphere collision algorithm (billiard ball model) to account for particle-particle and particle-wall collisions. All collisions are assumed to be binary and quasi-instantaneous, and with contact occurring at a single point. The algorithm considers impulse forces and momentum experienced by particles during collision and it also accounts for energy dissipation.

The motions of two particles at the time of the collision are expressed as:

$$\begin{aligned} m_1(V_1 - V_1^0) &= J \\ m_2(V_2 - V_2^0) &= -J \\ \frac{I_1}{r_1}(\omega_1 - \omega_1^0) &= J \times n \\ \frac{I_2}{r_2}(\omega_2 - \omega_2^0) &= J \times n \end{aligned} \quad (2.5)$$

where,

1 and 2 subscripts refer to particles participating in the collision

$m$  is the particle mass

$I$  is the particle moment of inertia

$V$  and  $\omega$  are the linear and angular particle velocities, respectively

$0$  superscript refers to particle velocities before the collision

$r$  is the particle radius

$J$  is the impulse force

$n$  is the unit vector in the normal direction

The impulse force in the normal direction is expressed by:

$$J_n = -(1 + e_n) \frac{V_{12}^0 \cdot n}{B} \quad (2.6)$$

The impulse force in the tangential direction for sticking collision is expressed as:

$$J_{t, sticking} = -(1+e_t) \frac{V_{12}^0 \cdot t}{B} \quad (2.7)$$

The impulse force in the tangential direction for sliding collision is expressed as:

$$J_{t, sliding} = -\mu_f J_n \quad (2.8)$$

In the above equations:

$e_n$  and  $e_t$  are the normal and tangential coefficients of restitution, respectively

$\mu_f$  is the friction coefficient

$t$  is the unit vector in the tangential direction

$B=1/m_1+1/m_2$

$V_{12}^0$  is the relative surface velocity (rotational and translational) of the two particles

## 2.4. Field Forces

The gravity and Buoyancy forces are automatically accounted for in the ANSYS Fluent MPM model.

Other particle-particle field forces, such as electrostatic, magnetic or cohesive forces, are implemented in the MPM using a potential force model in which inter-particle field forces acting on the particle are defined as:

$$F_i = \sum_j \frac{G_p M_i^{n1} M_j^{n2}}{R_{i-j}^{n3}} \quad (2.9)$$

where,

$M_i$  and  $M_j$  = masses of the interacting particles  $i$  and  $j$

$R_{i-j}$  = distance between the interacting particles  $i$  and  $j$

$G_p$ ,  $n_1$ ,  $n_2$ , and  $n_3$  = user-specified particle constants

The model constants  $G_p$ ,  $n_1$ ,  $n_2$ , and  $n_3$  can be indirectly related to the tensile strength or cohesive forces in solids.

In a similar manner, particle-wall field forces are defined as:

$$F_{i, wall} = \sum_{walls} \frac{G_w M_i^{n4}}{R_i^{n5}} \quad (2.10)$$

where,

$R_i$  = the closest distance of the particle from the wall  $w$

$G_w$ ,  $n_4$ ,  $n_5$ , = user-specified particle constants

The signs of the constants  $G_p$  and  $G_w$  determine whether the particle-particle and particle-wall field forces are attraction or repulsion forces.

## 2.5. Particle Deposition and Buildup

For filtration/separation-type applications, phenomena of particle deposition and buildup on selected surfaces are implemented based on the critical impact velocity algorithm. If the particle velocity is below the user-specified critical impact velocity, the particle will adhere to the wall upon impact. If the particle velocity is above the critical impact velocity, the particle will rebound off the wall. The deposited particles are assigned zero velocities and accelerations.

The deposition model also allows for detachment of the particle if the fluid force exceeds a critical user-specified limit.



---

## Chapter 3: Using the Macroscopic Particle Model

---

This chapter provides an overview of how to use the MPM add-on module in ANSYS Fluent. For more information about the theoretical background for the MPM modeling approach, see [Macroscopic Particle Model Theory \(p. 199\)](#).

This section is organized as follows:

- 3.1. Overview and Limitations
- 3.2. Loading the MPM add-on Module
- 3.3. Setting up MPM Model Simulations
- 3.4. Modeling Macroscopic Particles

### 3.1. Overview and Limitations

The MPM is an add-on module based on a suite of UDF functions that models transport of large particles in a fluid flow, particle-particle and particle-wall interactions, as well as particle tracking. The MPM is designed for macroscopic particles that are well-resolved across a particle diameter by at least 20-30 fluid cells.

The MPM is not a general purpose model. The model has been shown to give good results for applications in which macroscopic particles with Reynolds number  $Re_p \ll 1$  move in a flow having a fluid-to-particle density ratio,  $\rho_f / \rho_p$ , in the order of 1. For other values of  $Re_p$  and  $\rho_f / \rho_p$ , the drag prediction may not be accurate; in such cases the MPM model may be used when drag is not important.

Note the following limitations when using the MPM:

- Mass transfer and radiation cannot be modeled.
- The MPM is not compatible with moving/deforming meshes.
- Simulations of densely-packed particles are not supported because only one collision event is handled for each particle time step.
- There is no direct interaction with DPM particles (collision).
- Coupling with Multiphase models is possible, but manual settings are required for source terms (mass, momentum, and heat).
- Sub-iterations of MPM particle tracks within one time step are not supported.
- The MPM is not compatible with mesh interfaces.

### 3.2. Loading the MPM add-on Module

The MPM add-on module is installed with the standard installation of ANSYS Fluent in a directory called `addons/mpm`. The MPM add-on module is loaded into ANSYS Fluent through the text user interface

(TUI). Note that the module can be loaded only when a valid ANSYS Fluent case file has been set or read. To load the MPM add-on module, issue the following TUI command in the Fluent console:

```
define → models → addon-module
```

A list of ANSYS Fluent add-on modules is displayed in the console:

```
Fluent Addon Modules:  
0. none  
1. MHD Model  
2. Fiber Model  
3. Fuel Cell and Electrolysis Model  
4. SOFC Model with Unresolved Electrolyte  
5. Population Balance Model  
6. Adjoint Solver  
7. Single-Potential Battery Model  
8. Dual-Potential MSMD Battery Model  
9. PEM Fuel Cell Model  
10. Macroscopic Particle Model  
Enter Module Number: [0] 10
```

Select the Macroscopic Particle Model by entering the module number 10. During the loading process, a scheme library containing the graphical and text user interface, and a UDF library containing a set of user-defined functions (UDFs) for the MPM add-on module are automatically loaded into ANSYS Fluent. This process is reported in the Fluent console. Once the MPM module is loaded into ANSYS Fluent, the **Macroscopic Particles** model appears under the **Models** tree branch, and the UDF library also becomes visible as a new entry in the **UDF Library Manager** dialog box. By default, the **Macroscopic Particles** model is enabled.

---

### Note

The MPM model is only available in 3D.

---

## 3.3. Setting up MPM Model Simulations

The following procedure provides an overview of the steps required for setting up and solving an MPM model case. Only the steps that are pertinent to MPM modeling are shown here. For information about inputs related to other models that you want to use in conjunction with the MPM model, see the appropriate sections for those models in the [ANSYS Fluent User's Guide](#).

1. Use Fluent Launcher to start the **3D** version of ANSYS Fluent.
2. Read the case or mesh file.
3. Scale the mesh, if necessary.
4. For most cases except for cemented particles simulations, select the **Transient** solver from the **Time** list in the **General** task page.
5. Define the boundary conditions and physical properties for fluid flow as usual.
6. (optional) Initialize the flow variables and run the flow simulation until the flow converges.

This step could be performed to improve solution stability. Using a converged flow solution as a starting point for an MPM simulation can be particularly helpful in situations where there is a strong influence of macroscopic particles on the fluid flow.

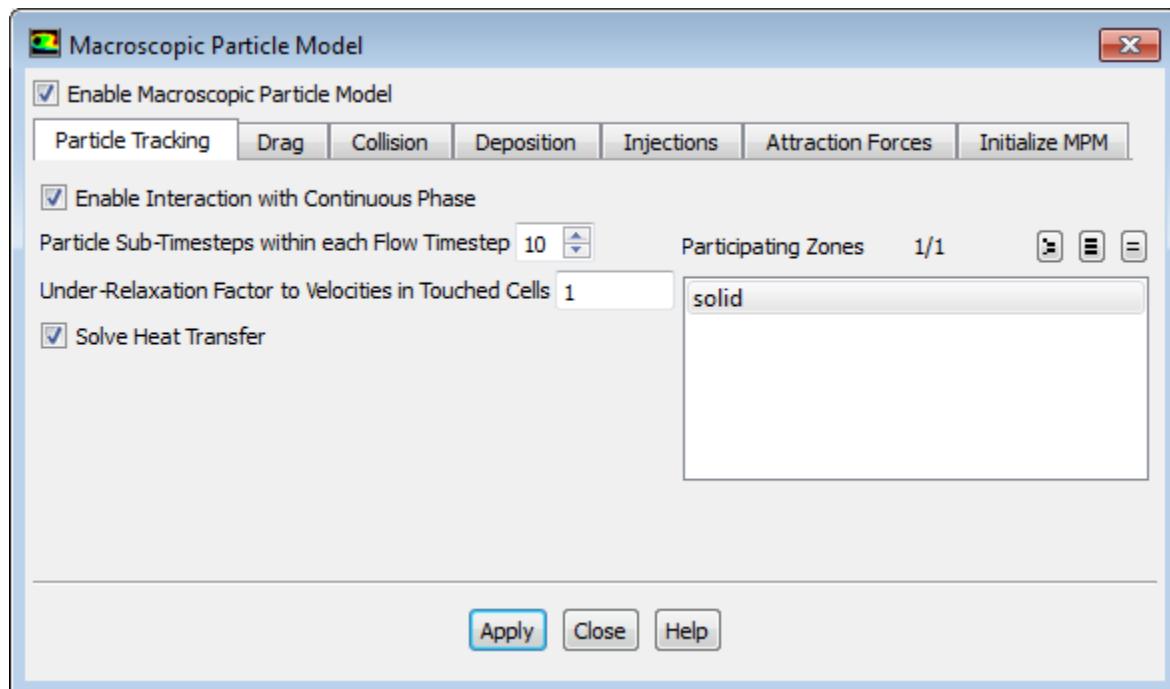
7. Use the **Macroscopic Particle Model** dialog box to specify the parameters for the MPM simulation.
8. Initialize either the MPM model as described in the sections that follow or the flow field in the usual way.
9. Run the MPM simulation.
10. Save the case and data files, if desired.
11. Review the simulation results by generating plots or alphanumeric reports using the ANSYS Fluent post-processing facilities.

## 3.4. Modeling Macroscopic Particles

You can set the MPM model parameters and select the options that are appropriate for your simulation using the **Macroscopic Particle Model** dialog box that opens from the **Setup/Models/Macroscopic Particle** tree item. From the **Macroscopic Particle Model** dialog box, you can access the **Create/Modify Injection** dialog boxes, where you can set the macroscopic particle injection properties.

 **Setup** → **Models** → **Macroscopic Particle**  **Edit...**

**Figure 3.1: Macroscopic Particle Model Dialog Box (Particle Tracking Tab)**



The inputs for the MPM model are entered using the following tabs:

### Particle Tracking

is where you can define parameters for particle tracking.

### Drag

contains drag law options and parameters for the particle drag calculations.

### Collision

allows you to enable particle-particle and particle-wall collisions and define the collision parameters.

### **Deposition**

allows you to enable particle deposition on selected surfaces and particles and define the relevant parameters.

### **Injections**

allows you to define macroscopic particle injections.

### **Attraction Forces**

is where you can specify parameters for particle-particle and particle-wall attraction forces.

### **Initialize MPM**

contains controls for initializing the macroscopic particles and displaying particle injections.

---

### **Note**

The default settings may not be appropriate for your case. You must provide parameter data for your specific material.

---

For additional information, see the following sections:

- 3.4.1. Specifying Particle Tracking Parameters
- 3.4.2. Specifying the Drag Law
- 3.4.3. Defining Parameters for Particle-Particle and Particle-Wall Collisions
- 3.4.4. Specifying Deposition Parameters
- 3.4.5. Specifying Injection Parameters
- 3.4.6. Defining Field Forces
- 3.4.7. Initializing the MPM model

## **3.4.1. Specifying Particle Tracking Parameters**

You can use the **Particle Tracking** tab to control the solution process of the MPM trajectory equations using the following settings (see [Figure 3.1: Macroscopic Particle Model Dialog Box \(Particle Tracking Tab\) \(p. 207\)](#)):

### **Particle Zones**

contains a selectable list of the cell zones in which you can use the MPM model.

### **Enable Interaction with Continuous Phase**

(if selected) enables two-way coupling between particles and fluid flow.

### **Particle Sub-Timesteps within each Flow Timetep**

is a number of sub-time steps for particle tracking in each flow time step.

### **Under-Relaxation Factor to Velocities in Touched Cells**

By default the under-relaxation factor for fixing velocities is set to 1.

## Solve Heat Transfer

enables solving the energy equation. When this option is enabled, you will need to specify **Initial Temperature** and **Specific Heat** for a particle material in the **Create/Modify Injection** dialog box.

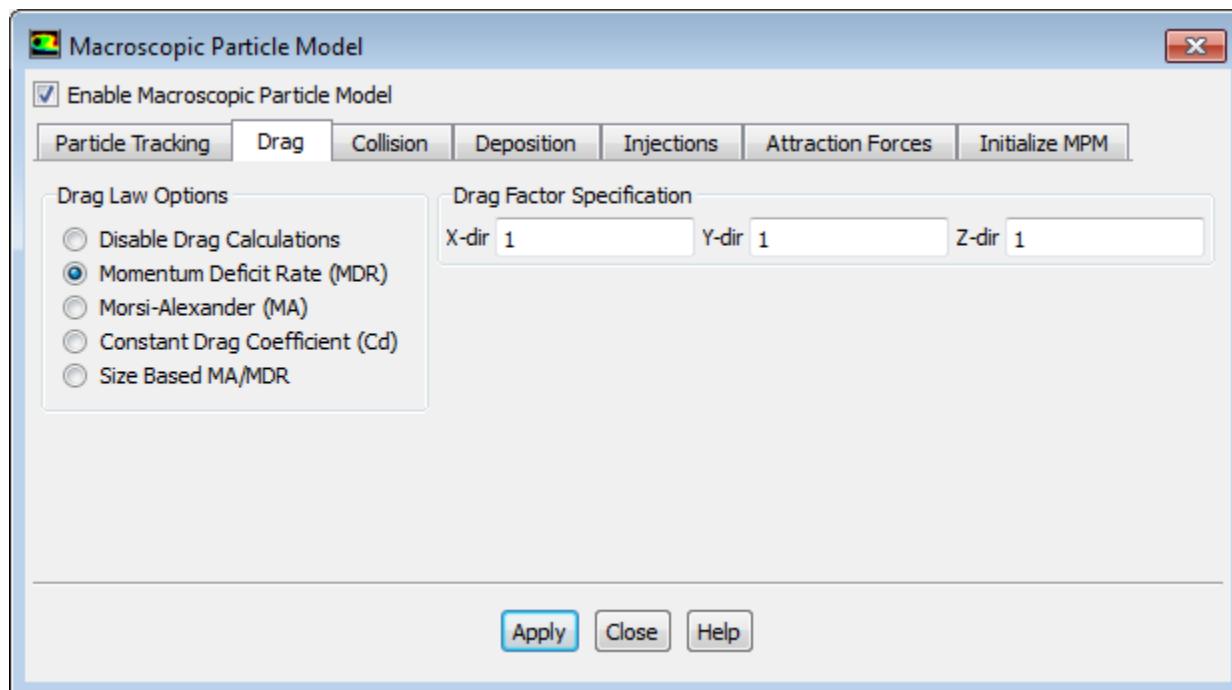
### Note

The particle time step should be chosen so that macroscopic particles do not cross more than one CFD mesh cell.

## 3.4.2. Specifying the Drag Law

Under the **Drag** tab, you can select an appropriate for your simulation drag law and set the relevant parameters or select **Disable Drag Calculations** if you do not want to model the drag force in the domain.

**Figure 3.2: Macroscopic Particle Model Dialog Box (Drag Tab)**



In the ANSYS Fluent MPM model, the following formulations are available for modeling the drag forces between the continuous and macroscopic particle phases:

- **Momentum Deficit Rate (MDR)** (default)

In this formulation, the momentum exchange coefficient between fluid and solid phases  $K_{ls}$  is calculated as:

$$K_{ls} = f / \Delta t$$

where  $f$  is the drag function, and  $\Delta t$  is the macroscopic particle time step.

The accuracy of the implicit calculations using the Momentum Deficit Rate model depends on number of cells comprising the macroscopic particle. This drag correlation is more suitable for bigger size particles.

- **Morsi-Alexander (MA)**

In the Morsi-Alexander based drag correlation [3] (p. 225) adopted in ANSYS Fluent, the Reynolds number for the fluid phase is calculated by:

$$Re = \rho_f d_p / \mu_f v_{fp}$$

where,

$\rho_f$  = fluid density

$\mu_f$  = fluid viscosity

$d_p$  = particle diameter

$v_{fp}$  = relative velocity of the discrete and fluid phases

The Morsi-Alexander based drag correlation is suitable for smaller size particles.

- **Constant Drag Coefficient (Cd)**

This option implies that the drag is evaluated using a constant drag coefficient. This correlation is suitable for cases where the particle is moving at a certain specified velocity (or acceleration).

- **Size Based MA/MDR**

When this option is selected, the ANSYS Fluent solver automatically switches between the Momentum Deficit Rate and Morsi-Alexander drag laws based on a critical diameter. This drag formulation is best suited for cases involving wide range of particle sizes in a single simulation. For particles with a diameter below the value that you specify for **Critical Diameter**, the Mosi-Alexander drag law will be used. Otherwise, the Momentum Deficit Rate formulation will be used.

You can define a drag factor in a specific direction (**X-dir,Y-dir,Z-dir**). A drag factor of 1 means that the virtual mass force equalizes the momentum difference of fluid in cells touched by the particle surface within a single flow time step. As a consequence, the particle sub-time step has to be much smaller (about one-fourth) than the momentum relaxation time  $\tau_\infty$  [5] (p. 225):

$$\tau_\infty = \frac{(\rho_p + \rho_f)/2 d_p^2}{18\mu}$$

where  $\rho_p$  and  $\rho_f$  are the particle and fluid densities, respectively,  $d_p$  is the particle diameter, and  $\mu$  is the fluid viscosity.

Besides, the accuracy of the predicted drag depends on the number of cells touched by the particle. At least 20-30 cells are required across the particle diameter.

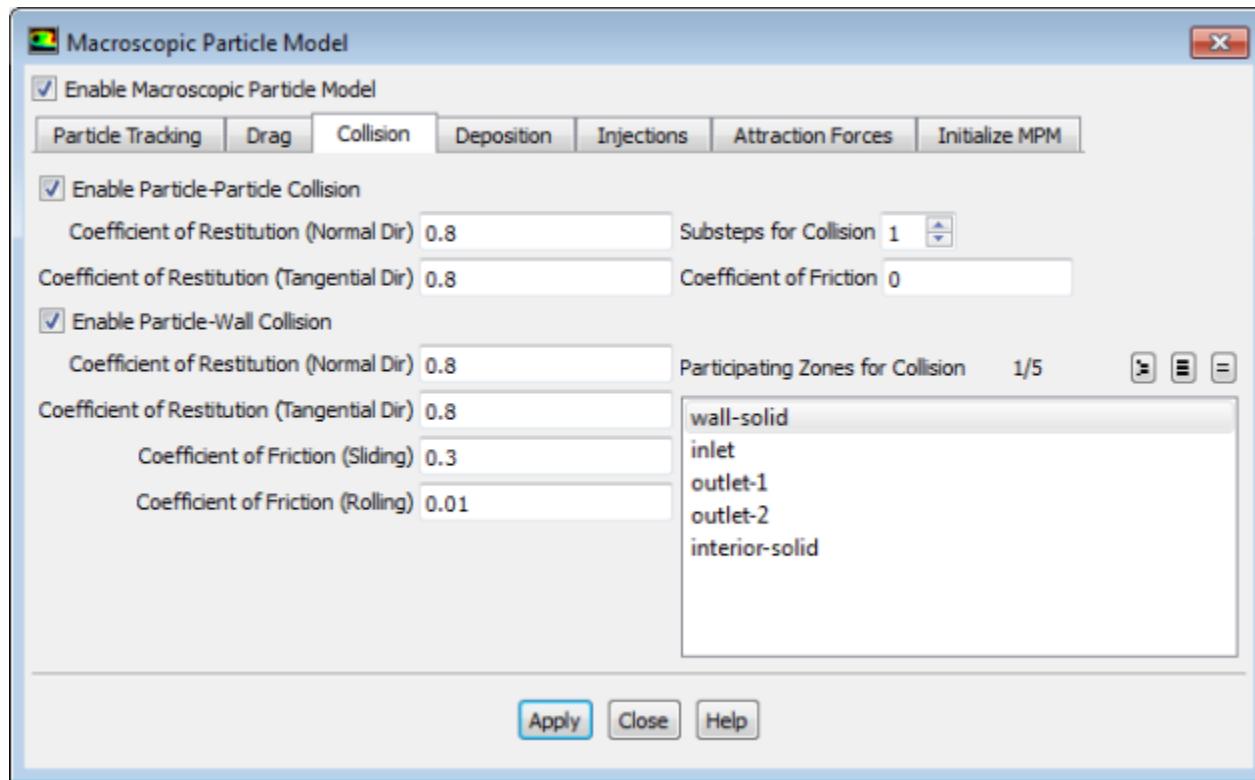
For multiphase flows, you can select an appropriate fluid velocity field to be used in the drag force calculations. The following options are available in the **Multipahse: Drag Based On** group box:

- **Primary Phase Velocity**
- **Secondary Phase Velocity**
- **Volume-Weighted Mixture Velocity**

### 3.4.3. Defining Parameters for Particle-Particle and Particle-Wall Collisions

The **Collision** tab allows you to model the effects of particle-particle and particle-wall collisions.

**Figure 3.3: Macroscopic Particle Model Dialog Box (Collision Tab)**



To account for collisions between particles:

1. Select **Enable Particle-Particle Collision**.
2. Specify the following parameters:

**Coefficient of Restitution (Normal Dir)**

corresponds to  $e_n$  in [Equation 2.6 \(p. 201\)](#).

**Coefficient of Restitution (Tangential Dir)**

corresponds to  $e_t$  in [Equation 2.7 \(p. 202\)](#).

**Substeps For Collision**

is a number of sub-time steps for detecting particle collision in each particle tracking time step.

**Coefficient of Friction**

corresponds to  $\mu_f$  in [Equation 2.8 \(p. 202\)](#).

If you also want to account for collisions between particles and walls:

1. Select **Enable Particle-Wall Collision**.
2. From the **Participating Zones For Collision** selection list, select the appropriate walls.

3. In addition to **Coefficient of Restitution (Normal Dir)** and **Coefficient of Restitution (Tangential Dir)** on the selected walls, you can specify the following parameters:

#### **Coefficient of Friction (Sliding)**

corresponds to  $\mu_f$  in [Equation 2.8 \(p. 202\)](#).

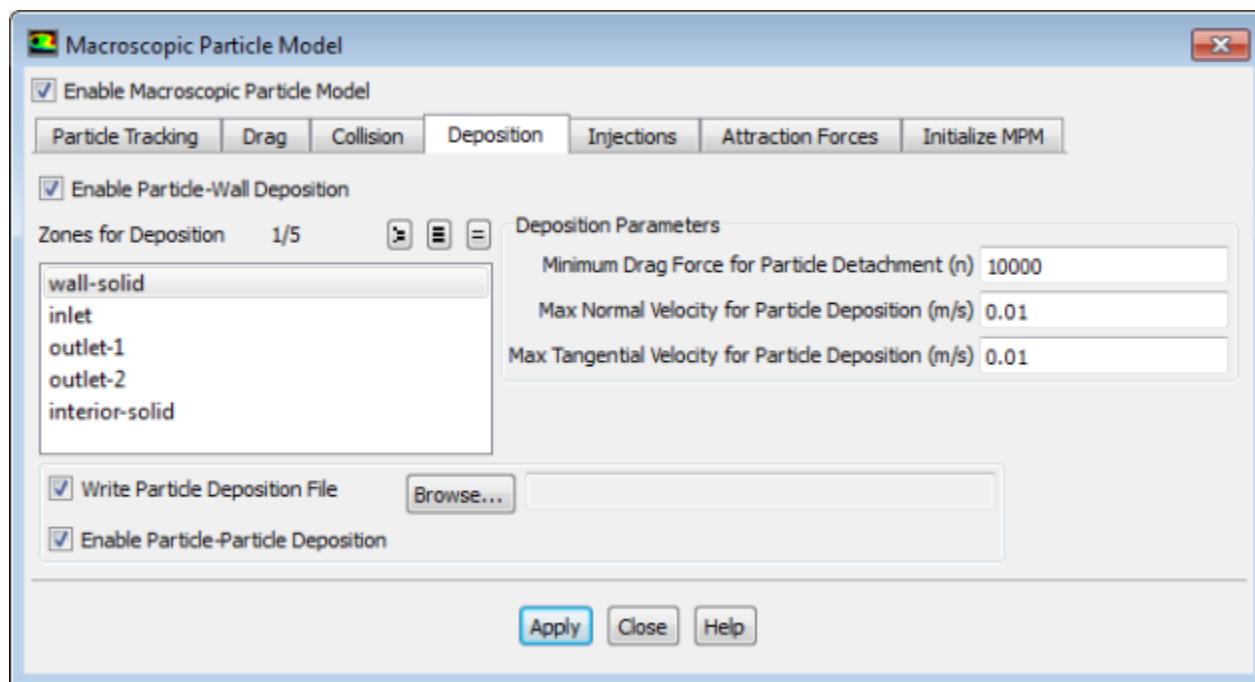
#### **Coefficient of Friction (Rolling)**

The value for the rolling coefficient of friction will be factored in the friction coefficient  $\mu_f$ .

### **3.4.4. Specifying Deposition Parameters**

You can use the **Deposition** tab to specify parameters for macroscopic particle deposition and buildup on selected zones.

**Figure 3.4: Macroscopic Particle Model Dialog Box (Deposition Tab)**



1. Select **Enable Particle-Wall Deposition**.
2. From the **Zones for Deposition** selection list, select the wall zones on which the macroscopic particles will adhere upon collision.
3. Specify the following deposition parameters:

#### **Minimum Drag Force for Particle Detachment**

specifies the force at or above which the deposited particles detach from a surface zone.

#### **Max Normal Velocity for Particle Deposition, Max Tangential Velocity for Particle Deposition**

specify the normal and tangential components of the approach velocity at or below which the particles deposit on the surface.

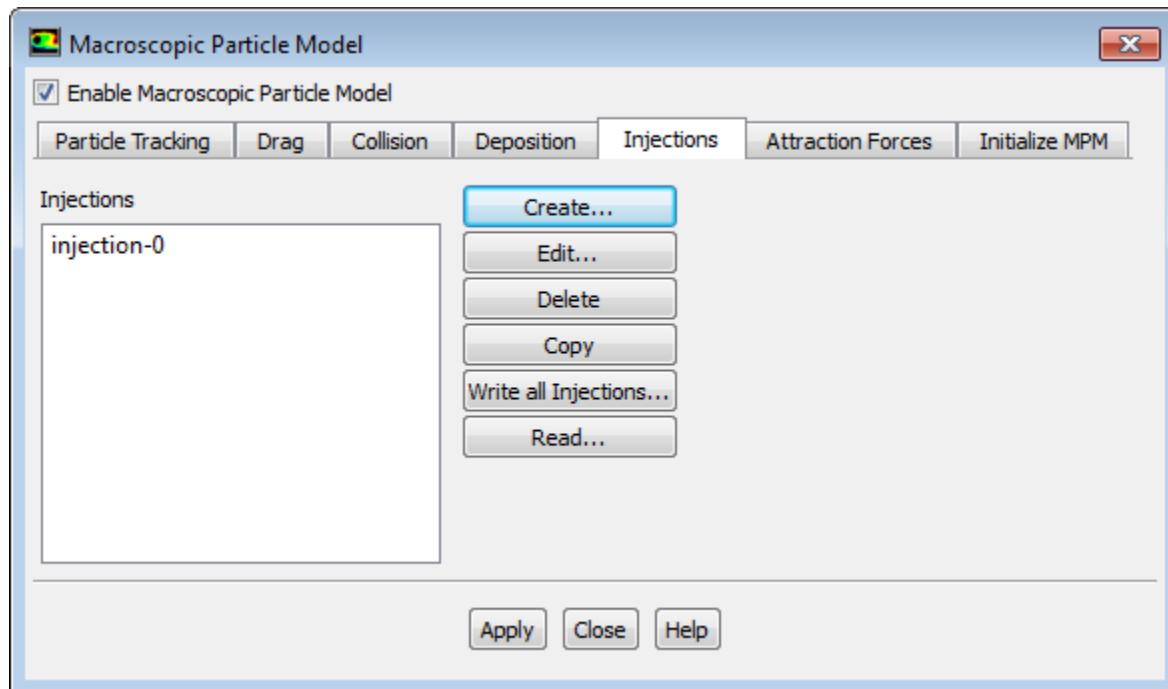
4. If you want to save particle deposition history data in ASCII format, enable **Write Particle Deposition File**.

- To account for particle agglomeration, select **Enable Particle-Particle Deposition**. A particle that collides with any previously deposited particles will attach to the particle surface. The deposition parameters that you have specified for particle-wall deposition will be also used for modeling particle-particle deposition.

### 3.4.5. Specifying Injection Parameters

You can use the **Injections** tab to create, edit, copy or delete macroscopic particle injections. You can also save all particle injections that you have created to a file for later use or read injections from a previously generated file. The controls under the **Injection** tab are similar to those found in the **Injections** dialog box (see [Injections Dialog Box in the \*Fluent User's Guide\*](#)) with a few minor differences.

**Figure 3.5: Macroscopic Particle Model Dialog Box (Injection Tab)**



The procedures for creating, modifying and interacting with injections are also similar to those for the DPM injections. The differences related to the MPM injections will be further emphasized. For more information about using these controls, see the [Fluent User's Guide](#).

- To create an MPM injection: click **Create...** and set the injection properties using the [Create/Modify Injection](#) dialog box (see [Defining MPM Injection Properties \(p. 214\)](#)). After the injection is created, it will appear in the **Injections** selection list, in the **Macroscopic Particle Model** dialog box.
- To edit an existing MPM injection: select it from the **Injections** selection list, click **Edit...**, and in the [Create/Modify Injection](#) dialog box that will open, modify the injection properties as necessary .
- To copy an existing MPM injection to a new injection: select it from the **Injections** selection list and click **Copy**. The name of the new copied injection appended with \_d will appear in the **Injections** list.
- To delete an existing MPM injection: select it from the **Injections** selection list and click **Delete**.

- To write information about all injections created in a case: click **Write All Injections...** and in the **Select File** dialog box that will open, enter the name for the injection file.
- 

#### Note

**Write All Injections...** is not supported for steady flows with cemented particles.

---

- To read previously generated injection file into your case: click **Read...** and in the **Select File** dialog box that will open, select the injection file to read in. If the injection with the same name already exists in your current case, then ANSYS Fluent will rename the imported injection by appending `_d` to the name.

### 3.4.5.1. Defining MPM Injection Properties

You can define the properties of an MPM injection using the **Create/Modify Injection** dialog box that can be accessed by clicking **Create....** The procedure for defining a particle injection is as follows:

1. (Optional) Change the default name of the injection (`injection-id`) by entering a new name in the **Injection Name** entry field.
  2. From the **Injection Type** drop-down list, select the injection type. In the ANSYS Fluent MPM model, you can create the following types of injections:
    - **point**: defines an injection released from a single point.
    - **plane**: defines uniformly or randomly distributed injections released from a plane.
    - **packing**: defines an injection of particles packed in a box or cylinder.
    - **from-file**: allows you to read a text file that defines an injection.
  3. If you want to model fixed particles, enable **Cemented Particle(s)**.
- 

#### Important

For steady-state simulations, all particles must be cemented.

---

4. Specify the particle physical parameters (**Diameter** and **Density**).
- 

#### Note

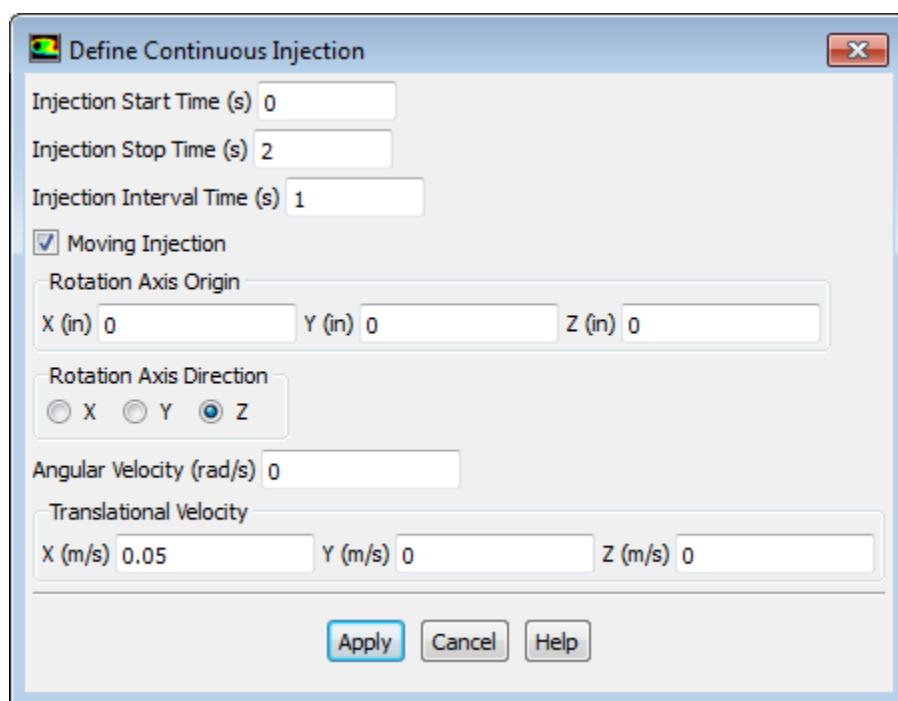
Note that a material defined in ANSYS Fluent cannot be assigned to a particle.

---

5. Specify the injection type-specific settings as described in the following sections:

- **point** injections: [Inputs for point Injections \(p. 216\)](#)
- **plane** injections: [Inputs for plane Injections \(p. 217\)](#)
- **packing** injections: [Inputs for packing Injections \(p. 219\)](#)

- from-file injections: [Inputs for from-file Injections \(p. 220\)](#)
6. (Optional) If you want to model a continuous generation of particle injections, select **Enable Continuous Injection** and in the **Define Continuous Injection** dialog box that opens when you click **Set...**, specify the relevant parameters.



You can specify the following settings:

- Injection start, stop, and interval times

New macroscopic particle will be injected into the domain at regular time intervals specified in **Injection Interval Time** from the **Injection Start Time** to the **Injection Stop Time**.

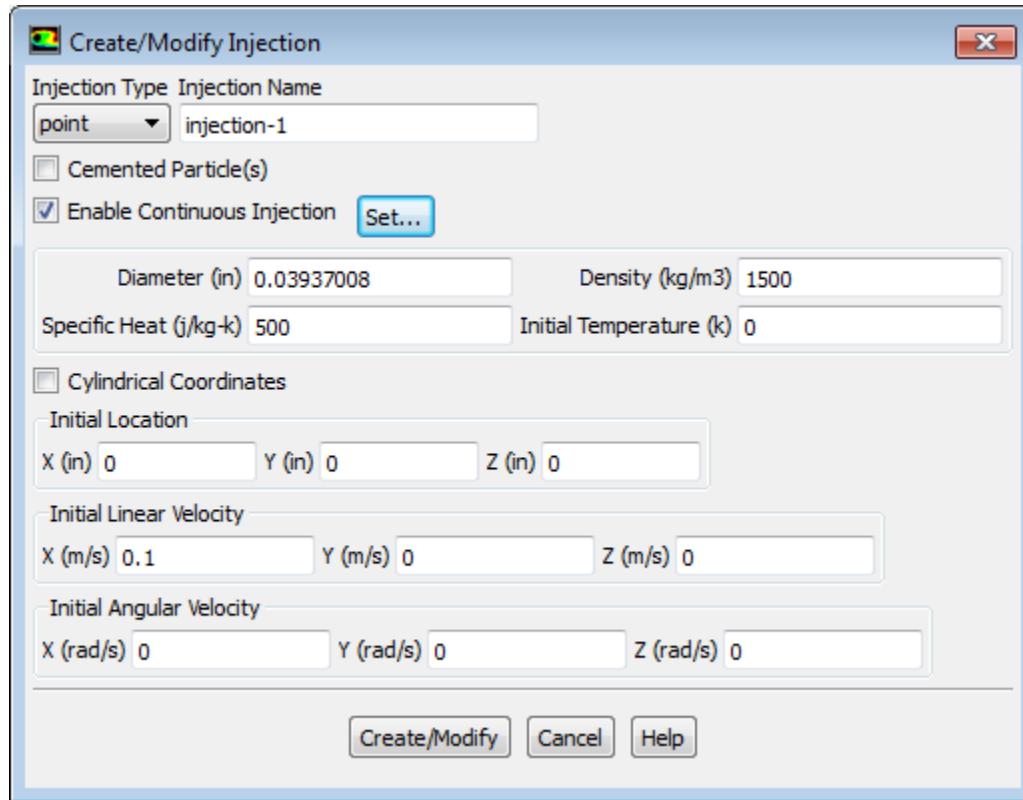
- (Optional) Moving and/or rotating injection release point parameters (available when **Moving Injection** is enabled):
  - The origin (**X**, **Y**, **Z**) and the direction of the axis (**X**, **Y**, or **Z**) about which the injection is rotating
  - The **Angular Velocity** of the injection location
  - The **X**, **Y**, **Z** components of the **Translational Velocity** of the injection location

At each time step, the MPM solver determines a new position of a particle injection based on the injection origin at the previous time step and the specified translational velocities.

## 7. Click **Create/Modify**.

The new injection that you have created appears under the **Injections** selection list.

### 3.4.5.2. Inputs for point Injections



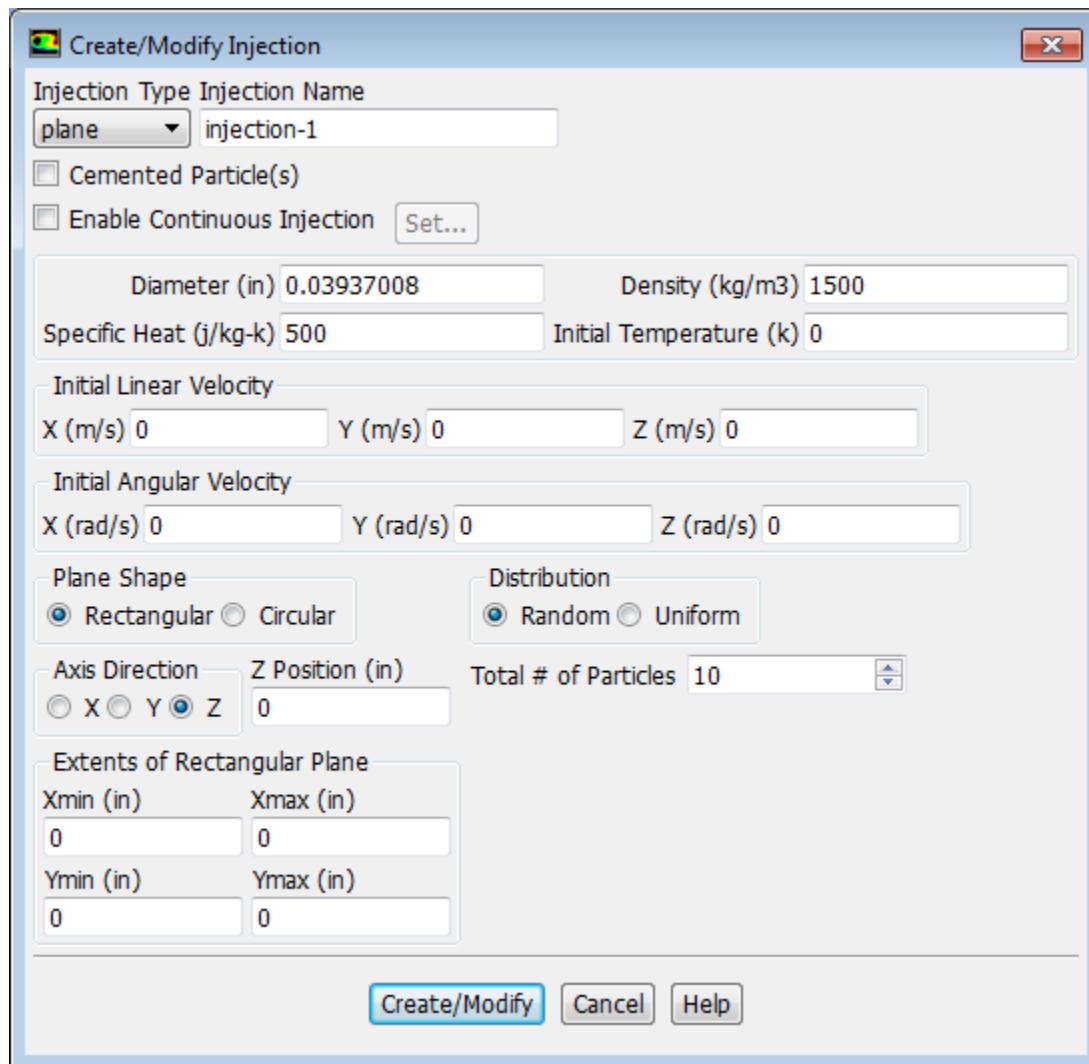
For a point injection of a macroscopic particle, you need to define the following initial conditions in the **Create/Modify Injection** dialog box:

- Particle diameter
- Particle density
- Specific heat of the particle material (only for heat transfer simulations)
- Initial particle temperature (only for heat transfer simulations)
- Initial particle release location:
  - Cartesian coordinates: **X**, **Y**, and **Z**
  - Cylindrical coordinates: **Radius**, **Angle**, and **Axial**
- Particle initial velocity:
  - Cartesian coordinates: **X**, **Y**, and **Z**
  - Cylindrical coordinates: **Radial**, **Tangential**, and **Axial**
- Particle initial angular velocity

## Cylindrical Coordinate System

By default, the global Cartesian coordinate system is used. If you want to specify injection parameters in a local cylindrical coordinate system, enable the **Cylindrical Coordinates** option, and then define the location of its origin (**X**, **Y**, **Z**) and **Axis Direction**.

### 3.4.5.3. Inputs for plane Injections



To define macroscopic particle injections released from a plane:

1. Specify the particle initial conditions as described in [Inputs for point Injections \(p. 216\)](#), except for the particle release location. Note that for plane injections, velocities and initial position can be specified only in Cartesian coordinates.
2. Define the macroscopic particle injection plane.

Two options are available:

- **Rectangular**

Macroscopic particles will be injected from a rectangular plane surface that is offset from a coordinate plane by a distance specified in **X**, **Y**, or **Z Position** in the selected **X**, **Y**, or **Z** axis direction.

The minimum and maximum limits of the rectangular surface are defined under the **Extents of Rectangular Plane** group box.

- **Circular**

Macroscopic particles will be injected from a circular plane surface that is perpendicular to a selected coordinate direction (**X**, **Y**, or **Z**). The origin and radius of the circle are specified under the **Center Location/Radius of Circular Plane** group box.

3. Define the distribution of the particle release locations.

Two modeling options are available:

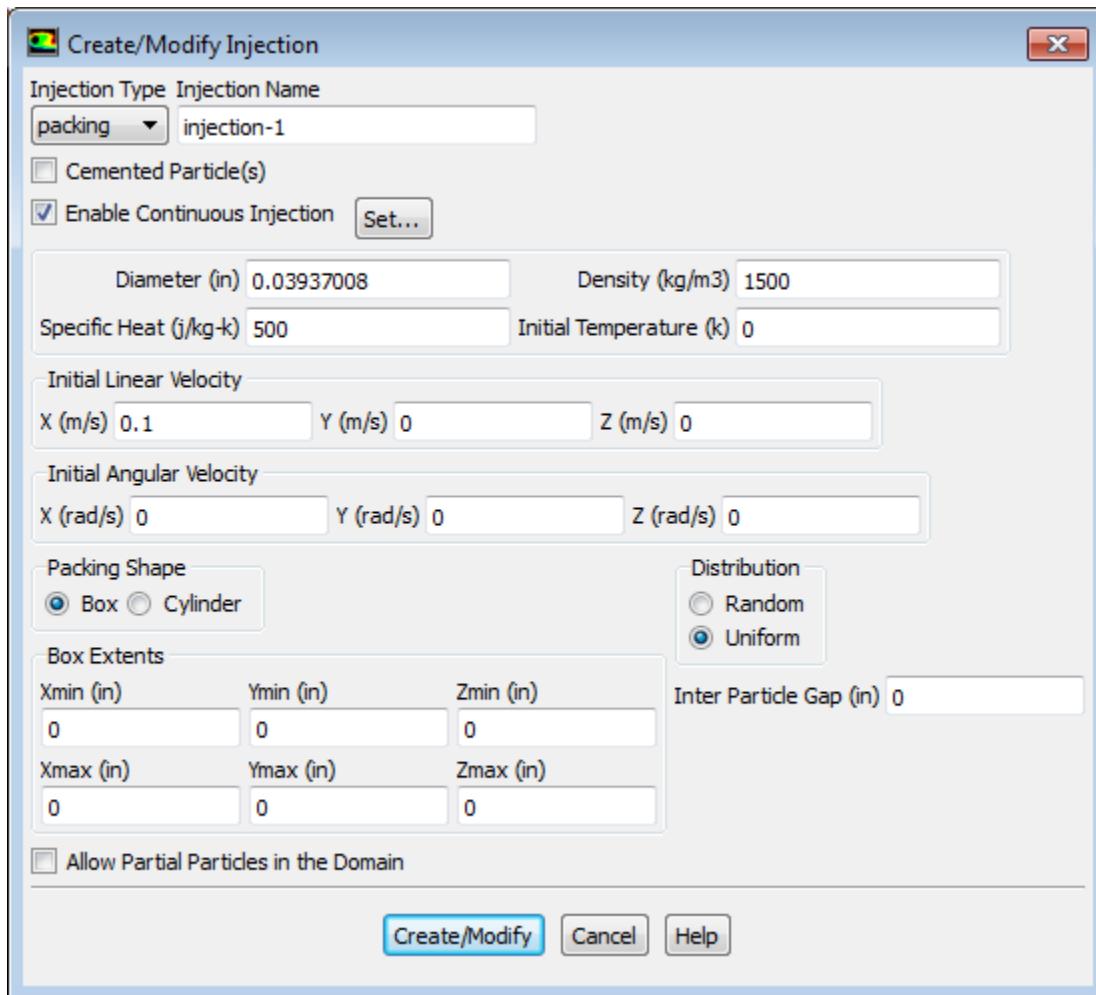
- **Random**

The MPM model will inject particles into a flow domain from random locations on the injection plane. The total number of the released particles is specified in the **Total # of Particles** integer entry field.

- **Uniform**

The particles will be injected into a flow domain from points arranged in a rectangular pattern. The number of the release points in two axial directions is specified in the appropriate integer entry fields (for example, **# of Particles in x-dir** and **# of Particles in y-dir**).

### 3.4.5.4. Inputs for packing Injections



To define an injection of macroscopic particles packed in a box or cylinder:

1. Specify particle initial conditions as described in [Inputs for point Injections \(p. 216\)](#), except for the particle release location. Note that for plane injections, velocities and initial position can be specified only in Cartesian coordinates.
2. Define the packing region.

For a **Box**-type packing, specify the minimum and maximum X, Y, and Z coordinates for the packing bounding box under the **Box Extents** group box.

For a **Cylinder**-type packing, specify the bounding cylinder axis, minimum and maximum coordinates for the cylinder height, cylinder origin and inner and out diameters.

3. Define the particle distribution in the packing.

Two modeling options are available:

- **Random**

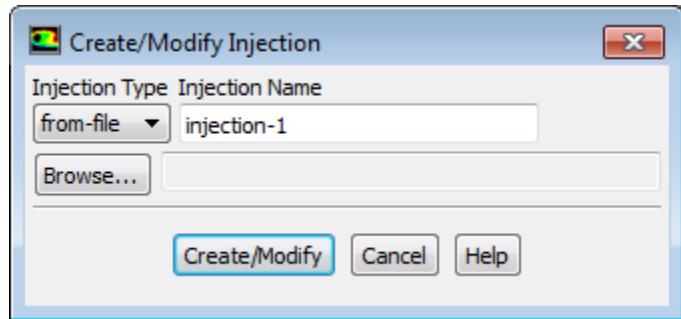
The initial volume fraction of particles in the packing is specified in the **Particle VOF** entry field.

- **Uniform**

The distance between particles in the initial spatially uniform packing is specified in the **Inter Particle Gap** entry field.

4. If you want to allow a part of the particle to be outside of the domain (with the particle center being in the domain), enable **Allow Partial Particles in the Domain**.

### **3.4.5.5. Inputs for from-file Injections**



For injections that you want to specify using the `from-file` option, the file format is:

```
number-of-particles
diameter density x-pos y-pos z-pos x-vel y-vel z-vel x-rot y-rot z-rot pstart pstop pinterval cemented
...

```

where the first line specifies the number of particles (`number-of-particles`), followed by `number-of-particles` lines that define parameters for the injection particles. For each particle, you must specify the diameter, density, initial position, linear and angular velocities, start, end, and interval times, and the integer flag (`cemented` in the above file format specification) indicating whether or not the particle is moving. Note that `cemented=1` indicates a stationary particle, while `cemented=0` indicates a moving particle. Values describing an injection should be separated by one or more spaces and can be specified using scientific notation (e or E) if needed.

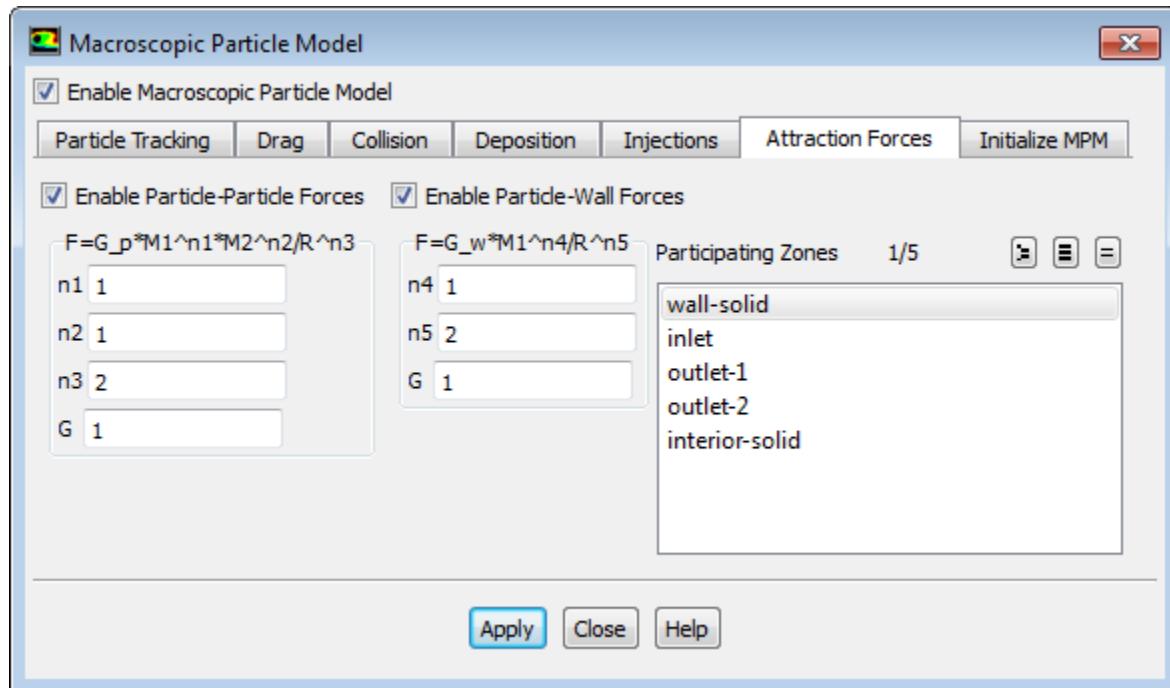
All quantities are in SI units.

An example is shown below:

```
1
5e-2 5.08e-2 -1e-8 1e-7 2.4e-1 -2e-4 1e-5 1.1 0.0 0.0 0.0 0.0 1e-2 1e-4 0
```

### **3.4.6. Defining Field Forces**

The **Attraction Forces** tab allows you to include attraction forces between particles and walls in your problem.

**Figure 3.6: Macroscopic Particle Model Dialog Box (Attraction Forces Tab)**

To define the attraction forces between macroscopic particles:

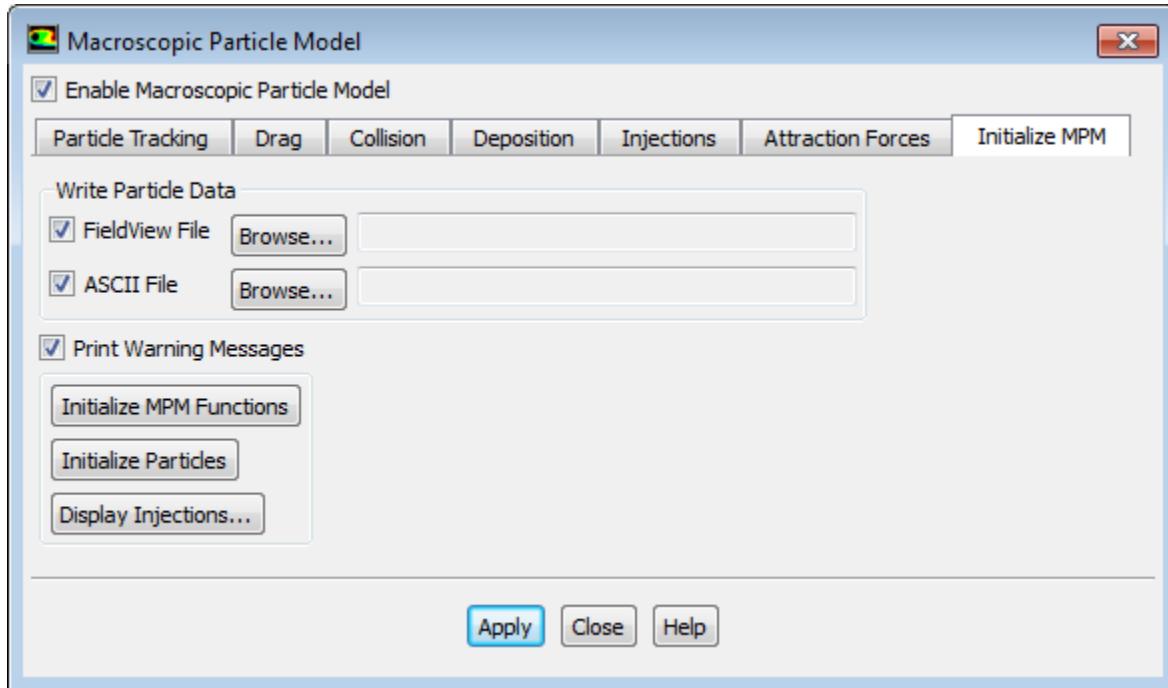
1. Select **Enable Particle-Particle Forces**.
2. Specify the model constants  **$n_1$ ,  $n_2$ ,  $n_3$ , and  $G_p$**  used in [Equation 2.9 \(p. 202\)](#).

To define the attraction forces between macroscopic particles and walls:

1. Select **Enable Particle-Wall Forces**.
2. From the **Participating Zones** selection list, select the
3. Specify the model constants  **$n_4$ ,  $n_5$ , and  $G_w$**  used in [Equation 2.10 \(p. 202\)](#).

### 3.4.7. Initializing the MPM model

The **Initialize MPM** tab allows you to apply macroscopic particle related source terms to the continuous phase, initialize macroscopic particles, and display particle trajectories.

**Figure 3.7: Macroscopic Particle Model Dialog Box (Initialize MPM Tab)**

You can also enable the following solution-related options:

- Saving particle data in a Field View and/or ASCII format
- Printing warning messages related to particle tracking in the Fluent console

Once you have set up the MPM model, you need to perform the following steps:

1. Click **Initialize MPM Functions**.

This will set up macroscopic particle related source terms and apply them to the continuous phase. It will also hook all appropriate UDF functions to the Fluent solver.

Initializing MPM functions adds the following two commands to the solver that will be executed during simulation:

- `mpm1`: displays the particle position(s) at each iteration step.
- `mpm3`: writes the particle position(s) at each iteration step in PNG format.

You can disable these commands or modify the reporting frequency in the **Execute Command** dialog box (accessible from the **Solution/Calculation Activities/Execute Commands** tree item).

If you want the solver to report or monitor additional solution quantities during calculation, you can define your own execute command(s) using Fluent TUI commands. For more information, see [Executing Commands During the Calculation in the \*Fluent User's Guide\*](#).

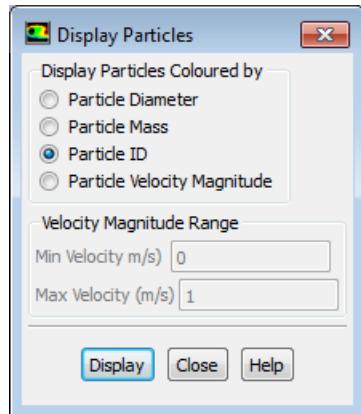
For coupled simulations, once you click **Initialize MPM Functions**, the pressure discretization scheme will automatically switch to **PRESTO!** to provide a more robust solution.

2. Initialize either the macroscopic particles by clicking **Initialize Particles** or the flow field in the usual way

The macroscopic particle(s) will be introduced into the fluid flow during the next computational time step.

3. Run the simulation with the injected macroscopic particle(s).

Upon completion of the MPM simulation, you can click **Display Injections...** and use the **Display Particles** dialog box to display the computed particle trajectories.



In the **Display Particles** dialog box, you can select the particle coloring option. You can color particle by the following particle variable values:

- diameter
- mass
- particle ID
- velocity magnitude

For the **Particle Velocity Magnitude** option, you can specify the minimum and maximum range of the velocity magnitude values.



---

# Bibliography

- [1] M. Agrawal, A. Bakker, and M. T. Prinkey. "*Macroscopic Particle Model – Tracking Big Particles in CFD*". *AIChE 2004 Annual Meeting*. Particle Technology Forum - Paper 268b. Austin, Texas, USA. 2004.
- [2] M. Agrawal, K. Ogawa. "*Drag force formulation in macroscopic particle model and its validation*". *AIChE 2009 Annual Meeting*. Paper 163b. Nashville, TN, USA. 2009.
- [3] S. A. Morsi and A. J. Alexander. "*An Investigation of Particle Trajectories in Two-Phase Flow Systems*". *J. Fluid Mech.* 55(2). 193–208. September 26 1972.
- [4] S. Ookawara, M. Agrawal, D. Street, and K. Ogawa. "*Quasi-direct numerical simulation of lift force-induced particle separation in a curved microchannel by use of a macroscopic particle model*". *Chemical Engineering Science*. 62. 9. 2454–2465. 2007.
- [5] S. Ookawara, M. Agrawal, D. Street, and K. Ogawa. "*Modeling the motion of a sphere falling in a quiescent Newtonian liquid in a cylindrical tube by using the macroscopic particle model*". *The Seventh World Congress of Chemical Engineering*. Glasgow, Scotland. C39-004. 2005.
- [6] D. Wadnerkar, M. Agrawal, and V. Pareek. "*Terminal Velocity of Particles Falling in Non-Newtonian Yield Pseudo-plastic Fluids using a Macroscopic Particle Model*". In *7th International Conference on Multiphase FLOW*. ICMF 2010Tampa, FL. 2010.



---

---

## **Part V: ANSYS Fluent Fuel Cell Modules**

[Using This Manual \(p. ccxxix\)](#)

[1. PEMFC Model Theory \(p. 231\)](#)

[2. Using the PEMFC Model \(p. 245\)](#)

[3. Fuel Cell and Electrolysis Model Theory \(p. 281\)](#)

[4. Using the Fuel Cell and Electrolysis Model \(p. 291\)](#)

[5. SOFC Fuel Cell With Unresolved Electrolyte Model Theory \(p. 327\)](#)

[6. Using the Solid Oxide Fuel Cell With Unresolved Electrolyte Model \(p. 337\)](#)

[Bibliography \(p. 355\)](#)

---

---

---

# Using This Manual

---

## 1. The Contents of This Manual

The ANSYS Fluent Fuel Cell Modules Manual provides information about the background and the usage of two separate add-on fuel cell modules for ANSYS Fluent. For each type of fuel cell add-on module, you will find background information pertaining to the models, a theoretical discussion of the models used in ANSYS Fluent, and a description of using the models for your CFD simulations. The available ANSYS Fluent add-on fuel cell modules are:

- PEMFC Model - allows you to model polymer electrolyte membrane fuel cells (PEMFC) with (or without) micro-porous layers. This model is a new and recommended model for simulating energy conversion processes in a PEM fuel cell. For more information, see [PEMFC Model Theory \(p. 231\)](#) and [Using the PEMFC Model \(p. 245\)](#).
- Fuel Cell and Electrolysis Model - allows you to model polymer electrolyte membrane fuel cells (PEMFC), solid oxide fuel cells (SOFC), and electrolysis with ANSYS Fluent. This model is sometimes referred to as the Resolved Electrolyte model. Note that the PEMFC sub-model is being retained in the current release. However, it will be removed in future releases because the new PEMFC Model mentioned above is more advanced and complete as described in the Release Notes for ANSYS Fluent R17.0. For more information, see [Fuel Cell and Electrolysis Model Theory \(p. 281\)](#) and [Using the Fuel Cell and Electrolysis Model \(p. 291\)](#).
- SOFC With Unresolved Electrolyte Model - allows you to model solid oxide fuel cells (SOFC). For more information, see [SOFC Fuel Cell With Unresolved Electrolyte Model Theory \(p. 327\)](#) and [Using the Solid Oxide Fuel Cell With Unresolved Electrolyte Model \(p. 337\)](#).

This part is divided into the following chapters:

- [PEMFC Model Theory \(p. 231\)](#)
- [Using the PEMFC Model \(p. 245\)](#)
- [Fuel Cell and Electrolysis Model Theory \(p. 281\)](#)
- [Using the Fuel Cell and Electrolysis Model \(p. 291\)](#)
- [SOFC Fuel Cell With Unresolved Electrolyte Model Theory \(p. 327\)](#)
- [Using the Solid Oxide Fuel Cell With Unresolved Electrolyte Model \(p. 337\)](#)
- [Bibliography \(p. 355\)](#)



# Chapter 1: PEMFC Model Theory

This chapter presents the theoretical background for the advanced Polymer Electrolyte Membrane Fuel Cell (PEMFC) modeling capabilities in ANSYS Fluent.

- 1.1. Introduction
- 1.2. Electrochemistry Modeling
- 1.3. Current and Mass Conservation
- 1.4. Water Transport and Mass Transfer in PEMFC
- 1.5. Heat Source
- 1.6. Properties
- 1.7. Transient Simulations
- 1.8. Leakage Current (Cross-Over Current)

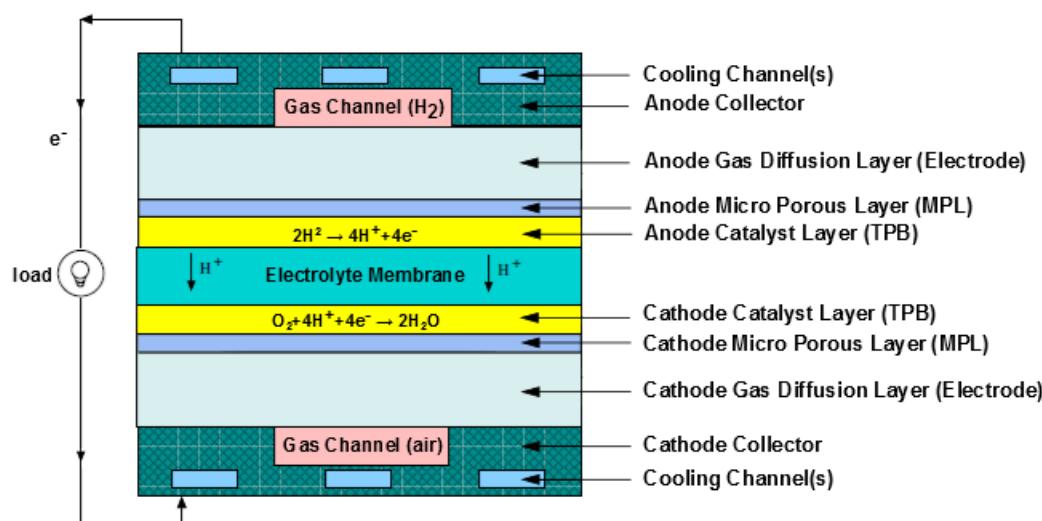
## 1.1. Introduction

The PEMFC module is provided as an add-on module with the standard ANSYS Fluent licensed software.

A fuel cell is an energy conversion device that converts the chemical energy of fuel into electrical energy. With the PEMFC model, both the Triple Phase Boundary (TPB), also known as the catalyst layer, and the ionic conducting electrolyte, also known as the membrane in PEMFC terminology, are included in the computational domain. The PEMFC module allows you to model polymer electrolyte membrane fuel cells.

To determine the physical domains that are included in the PEMFC module, a schematic of a polymer electrolyte membrane fuel cell (PEMFC) is shown in [Figure 1.1: Schematic of a PEM Fuel Cell \(p. 231\)](#).

**Figure 1.1: Schematic of a PEM Fuel Cell**

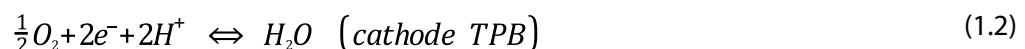


Hydrogen flows into the fuel cell on the anode side. It diffuses through the porous gas diffusion layer (GDL) and micro-porous layer (MPL), which is optional for the operation of a PEMFC, and then comes in contact with the catalyst layer. Here it forms hydrogen ions and electrons. The hydrogen ions diffuse

through the polymer electrolyte membrane at the center, the electrons flow through the gas diffusion layer to the current collectors and into the electric load attached. Electrons enter the cathode side through the current collectors and the gas diffusion layer. Similarly, oxygen (or air) flows into the fuel cell on the cathode side and diffuses through the porous gas diffusion layer and then micro-porous layer to reach the catalyst layer. At the catalyst layer, the electrons, the hydrogen ions, and the oxygen combine to form water.

In the PEMFC model in ANSYS Fluent, two electric potential fields are solved. One potential is solved in the membrane and the catalyst layer. The other is solved in the TPB catalyst layer, the micro-porous layer, the porous electrode, and the current collectors. The rates of electrochemical reactions are computed in the TPB layers at both the anode and the cathode. Based on the cell voltage that you prescribe, the current density value is computed. Alternatively, a cell voltage can be computed based on a prescribed average current density.

The polymer electrolyte membrane fuel cell (PEMFC) has emerged as a favored technology for auto transportation and power generation because it is compact, clean, runs at low temperature ( $<100^{\circ}\text{ C}$ ), permits an adjustable power output, and can be started relatively rapidly. Hydrogen is supplied at the anode and air is supplied at the cathode. The following electrochemical reactions take place in the anode and cathode triple phase boundary (TPB) layers, respectively,



Electrons produced in the anode travel through an external circuit to the cathode, while protons ( $\text{H}^+$ ) travel through the membrane from the anode TPB to the cathode TPB, thereby forming an electrical circuit.

In a PEM fuel cell, the three phases of water are present. The gas and liquid water phases are present in all the physical domains except the solid membrane and current collectors. Water is also present in the dissolved phase, but only inside the catalyst layers and the membrane. The water produced by the cathode side electro-chemistry is assumed to be in the dissolved phase ([Equation 1.2 \(p. 232\)](#)).

## 1.2. Electrochemistry Modeling

At the center of the electrochemistry is the computation of the rates of the anodic and cathodic reactions. The electrochemistry model adopted in ANSYS Fluent is one that has been used by other groups ([\[4\] \(p. 355\)](#), [\[5\] \(p. 355\)](#), and [\[12\] \(p. 355\)](#)).

The driving force behind these reactions is the surface overpotential: the difference between the phase potential of the solid and the phase potential of the electrolyte/membrane. Therefore, two potential equations are solved. One potential equation ([Equation 1.3 \(p. 232\)](#)) accounts for the electron transport of  $e^-$  through the solid conductive materials and is solved in the TPB catalyst layer, the solid grids of the porous media, and the current collector; the other potential equation ([Equation 1.4 \(p. 232\)](#)) represents the protonic (that is, ionic) transport of  $\text{H}^+$  and is solved in the TPB catalyst layer and the membrane. The two potential equations are as follows:

$$\nabla \cdot (\sigma_{sol} \nabla \phi_{sol}) + R_{sol} = 0 \quad (1.3)$$

$$\nabla \cdot (\sigma_{mem} \nabla \phi_{mem}) + R_{mem} = 0 \quad (1.4)$$

where

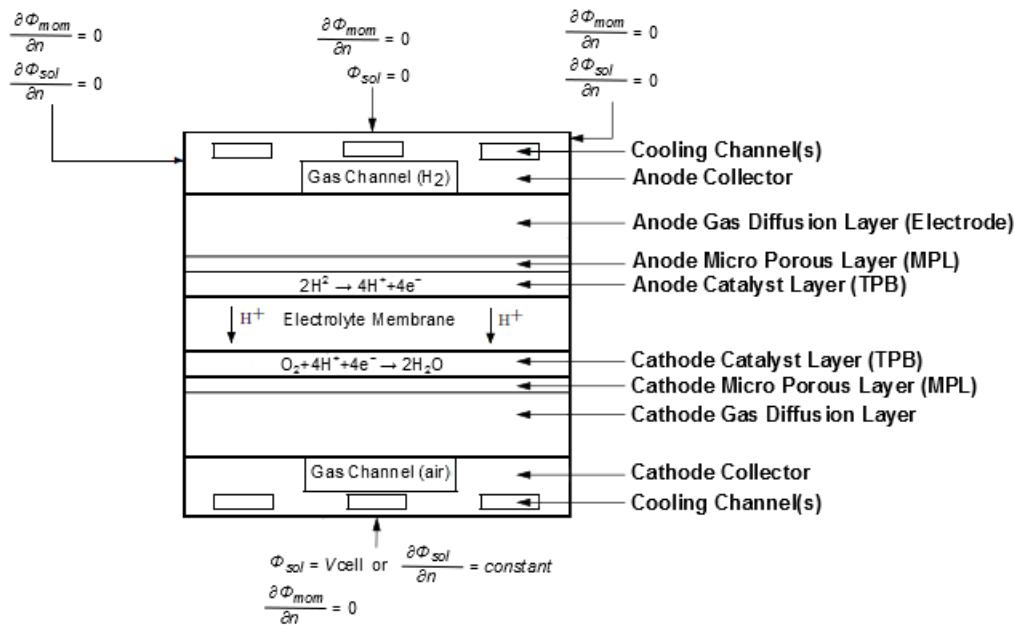
$\sigma$  = electrical conductivity (1/ohm-m)

$\phi$  = electric potential (volts)

$R$  = volumetric transfer current ( $A/m^3$ )

The following figure illustrates the boundary conditions that are used to solve for  $\phi_{sol}$  and  $\phi_{mem}$ .

**Figure 1.2: Boundary Conditions for the Electric Potentials (Solid and Membrane) — PEM Fuel Cell**



There are two types of external boundaries: those that have an electrical current passing through them, and those that do not.

As no ionic current leaves the fuel cell through any external boundary, there is a zero flux boundary condition for the membrane phase potential,  $\phi_{mem}$ , on all outside boundaries.

For the solid phase potential,  $\phi_{sol}$ , there are external boundaries on the anode and the cathode side that are in contact with the external electric circuit. Electrical current generated in the fuel cell only passes through these boundaries. On all other external boundaries there is a zero flux boundary condition for  $\phi_{sol}$ .

On the external contact boundaries, fixed values for  $\phi_{sol}$  (potentiostatic boundary conditions) are recommended. If the anode side is set to zero, the (positive) value prescribed on the cathode side is the cell voltage. Specifying a constant flux (say on the cathode side) means to specify galvanostatic boundary conditions.

The transfer currents, or the source terms in [Equation 1.3 \(p. 232\)](#) and [Equation 1.4 \(p. 232\)](#), are nonzero only inside the catalyst layers and are computed as:

- For the potential equation in the solid phase,  $R_{sol} = -R_{an} (< 0)$  on the anode side and  $R_{sol} = +R_{cat} (> 0)$  on the cathode side.

- For the potential equation in the membrane phase,  $R_{mem}=+R_{an}(>0)$  on the anode side and  $R_{mem}=-R_{cat}(<0)$  on the cathode side.

The source terms in [Equation 1.3 \(p. 232\)](#) and [Equation 1.4 \(p. 232\)](#), also called the exchange current density ( $A/m^3$ ), have the following general definitions:

$$R_{an} = \left( \zeta_{an} j_{an}(T) \right) \left( \frac{[A]}{[A]_{ref}} \right)^{\gamma_{an}} \left( e^{\alpha_{an} F \eta_{an}/RT} - e^{-\alpha_{cat} F \eta_{an}/RT} \right) \quad (1.5)$$

$$R_{cat} = \left( \zeta_{cat} j_{cat}(T) \right) \left( \frac{[C]}{[C]_{ref}} \right)^{\gamma_{cat}} \left( -e^{\alpha_{an} F \eta_{cat}/RT} + e^{-\alpha_{cat} F \eta_{cat}/RT} \right) \quad (1.6)$$

where

$j(T)$  = reference exchange current density per active surface area ( $A/m^2$ )

$\zeta$  = specific active surface area (1/m)

$[ ], [ ]_{ref}$  = local species concentration, reference value (kmol/m<sup>3</sup>)

$\gamma$  = concentration dependence

$\alpha_{an}^{an}$  and  $\alpha_{cat}^{an}$  = anode and cathode transfer coefficients of the anode electrode, respectively (dimensionless)

$\alpha_{an}^{cat}$  and  $\alpha_{cat}^{cat}$  = anode and cathode transfer coefficients of the cathode electrode, respectively (dimensionless)

$\eta_{an}$  = surface overpotential given by [Equation 1.11 \(p. 235\)](#)

$\eta_{cat}$  = surface overpotential given by [Equation 1.12 \(p. 235\)](#)

$F$  = Faraday constant ( $9.65 \times 10^7$  C/kmol)

$R$  = the universal gas constant

$T$  = temperature

The above equation is the general formulation of the Butler-Volmer function. A simplification to this is the Tafel formulation given by:

$$R_{an} = \zeta_{an} j_{an}(T) \left( \frac{[A]}{[A]_{ref}} \right)^{\gamma_{an}} \left( e^{\alpha_{an} F \eta_{an}/RT} \right) \quad (1.7)$$

$$R_{cat} = \zeta_{cat} j_{cat}(T) \left( \frac{[C]}{[C]_{ref}} \right)^{\gamma_{cat}} \left( e^{-\alpha_{cat} F \eta_{cat}/RT} \right) \quad (1.8)$$

By default, the Butler-Volmer function is used in the ANSYS Fluent PEMFC model to compute the transfer currents inside the catalyst layers. When the magnitude of the surface over-potential ( $\eta$ ) is large, the Butler-Volmer formulation reduces to the Tafel formulation.

In [Equation 1.5 \(p. 234\)](#) through [Equation 1.8 \(p. 234\)](#),  $[A]$  and  $[C]$  represent the molar concentration of the species upon which the anode and cathode reaction rates depend, respectively. That is,  $A$  represents  $H_2$  and  $C$  represents  $O_2$ .

The reference exchange current density  $j_{an}(T)$  and  $j_{cat}(T)$  are dependent on the local temperature as follows:

$$j_{an}(T) = j_{an}^{ref} e^{-E_{an}/RT(1-T/T_{an}^{ref})} \quad (1.9)$$

$$j_{cat}(T) = j_{cat}^{ref} e^{-E_{cat}/RT(1-T/T_{cat}^{ref})} \quad (1.10)$$

where

$T_{ref}$  = user-specified activation energy

$T_{ref}$  = user-specified reference temperature

$j_{an}^{ref}$  and  $j_{cat}^{ref}$  = reference exchange current density at a specified reference temperature

The driving force for the kinetics is the local surface overpotential,  $\eta$ , also known as the *activation loss*. It is generally the difference between the solid and membrane potentials,  $\phi_{sol}$  and  $\phi_{mem}$ .

$$\eta_{an} = \phi_{sol} - \phi_{mem} - U_{an}^0 \quad (1.11)$$

$$\eta_{cat} = \phi_{sol} - \phi_{mem} - U_{cat}^0 \quad (1.12)$$

The half cell potentials at anode and cathode  $U_{an}^0$  and  $U_{cat}^0$  are computed by Nernst equations as follows [10] (p. 355):

$$U_{an}^0 = E_{an}^0 - \frac{\Delta S_{an}}{2F} (T - T^0) - \frac{RT}{2F} \ln\left(\frac{p_{H_2}}{p^0}\right) \quad (1.13)$$

$$U_{cat}^0 = E_{cat}^0 + \frac{\Delta S_{cat}}{2F} (T - T^0) - \frac{RT}{2F} \ln\left(\frac{p_{H_2O}}{p_{sat} \sqrt{p_{O_2}/p^0}}\right) \quad (1.14)$$

where  $p_{sat}$  is the water saturation pressure (Equation 1.45 (p. 242)) and  $p_{H_2}$ ,  $p_{O_2}$ , and  $p_{H_2O}$  are the partial pressures of hydrogen, oxygen, and water vapor, respectively. In the above equations, the standard state ( $T^0$ ,  $p^0$ ), the reversible potentials  $E_{an}^0$  and  $E_{cat}^0$ , and the reaction entropies  $\Delta S_{an}$  and  $\Delta S_{cat}$  are user-specified quantities.

From Equation 1.3 (p. 232) through Equation 1.14 (p. 235), the two potential fields can be obtained.

### 1.2.1. The Cathode Particle Model

When Equation 1.8 (p. 234) is used to compute the cathode transfer current, the mass transport resistance in the catalyst microstructure is not considered ([10] (p. 355)). The resistance may consist of two parts:

- Resistance due to an ionomer film  $\mathfrak{R}_{ion}$
- Resistance due to a liquid water film surrounding particles  $\mathfrak{R}_{liq}$

In the ANSYS Fluent PEMFC model, including these resistances in calculations of the transfer current is optional. The volumetric transfer current inside the cathode layers is represented by:

$$R_{cat} = 4F \frac{c_{O_2}}{c_{O_2}/j_{O_2}^{ideal} + \mathfrak{R}_{ion} + \mathfrak{R}_{liq}} \quad (1.15)$$

where  $c_{O_2}$  is the concentration of oxygen at the wall. The  $\mathfrak{R}_{ion}$  is a user-specified value, and the  $\mathfrak{R}_{liq}$  is calculated by:

$$\mathfrak{R}_{liq} = \frac{\zeta_{cat} r_p^2}{K_w D_w} \cdot \frac{\sqrt[3]{1 + \frac{s\epsilon}{1-\epsilon}} - 1}{3(1-\epsilon)} \quad (1.16)$$

where

$\zeta_{cat}$  = specific active surface area for the cathode catalyst (1/m)

$s$  = liquid saturation

$\epsilon$  = porosity

$r_p$  = particle diameter

$K_w D_w$  = product of oxygen solubility and diffusivity in liquid water (on the order of  $10^{-10} \text{ m}^2/\text{s}$ )

The  $j_{O_2}^{ideal}$  in [Equation 1.15 \(p. 235\)](#) is calculated as:

$$j_{O_2}^{ideal} = \frac{R_{cat}^0}{4F} \quad (1.17)$$

Here,  $R_{cat}^0$  is the ideal transfer current computed using [Equation 1.6 \(p. 234\)](#), but without considering resistance.

### 1.3. Current and Mass Conservation

Volumetric source terms ( $\text{kg}/\text{m}^3\text{-s}$ ) for  $H_2$ ,  $O_2$ , and the dissolved water content  $\lambda$  in the triple-phase boundaries (catalyst layers) due to electrochemical reactions are:

$$S_{H_2} = -\frac{M_{w,H_2}}{2F} R_{an} < 0 \quad (1.18)$$

$$S_{O_2} = -\frac{M_{w,O_2}}{4F} R_{cat} < 0 \quad (1.19)$$

$$S_\lambda = \frac{M_{w,H_2O}}{2F} R_{cat} > 0 \quad (1.20)$$

In the above equations,  $M_{w,H_2O}$ ,  $M_{w,O_2}$ , and  $M_{w,H_2}$  are the molecular mass of water, oxygen and hydrogen, respectively,  $F$  is the Faraday constant, and 2 and 4 are the numbers of electrons per mole of reactants and products.

Since the total electrical current produced in the cathode and the anode catalyst layer, respectively, is the same, we have the following equation for current conservation:

$$\int_{anode} R_{an} dV = \int_{cathode} R_{cat} dV \quad (1.21)$$

### 1.4. Water Transport and Mass Transfer in PEMFC

As stated earlier, water is present in three phases in a PEM fuel cell. Depending on local thermodynamic and fluid dynamic conditions, mass transfer may occur between the three phases. For example, since PEM fuel cells operate under relatively low temperature ( $<100^\circ \text{ C}$ ), the water vapor may condense to liquid water, especially at high current densities. Dissolved water is generated by cathode-side reactions, and, depending on the local state of deviation from equilibrium, part of it may convert to either the liquid or gas phase. The dissolved phase can also be transported across the membrane from cathode to anode, or from anode to cathode.

While the presence of water keeps the membrane hydrated (which is necessary for PEM fuel cell operation), the liquid water blocks the gas diffusion passage and decreases the diffusion rate and the effective reacting surface area hence reducing the cell performance. Therefore, water formation and transport should be considered when modeling PEMFC systems. In this section, the modeling approaches adopted in ANSYS Fluent are described.

### 1.4.1. The Dissolved Phase Model

The dissolved phase exists in the catalyst layers (ionomers) and the membrane. The generation and transport of dissolved water is described by [13] (p. 355):

$$\frac{\partial}{\partial t} \left( \varepsilon_i M_{w,H_2O} \frac{\rho_i}{EW} \lambda \right) + \nabla \cdot \left( \vec{i}_m \frac{n_d}{F} M_w \right) = \nabla \cdot \left( M_w D_w^i \nabla \lambda \right) + S_\lambda + S_{gd} + S_{ld} \quad (1.22)$$

where

$\varepsilon_i$  = porosity of porous media

$\vec{i}_m$  = the ionic current density calculated as  $\vec{i}_m = -\sigma_{mem} \nabla \phi_{mem}$

$\lambda$  = dissolved water content

$n_d$  = osmotic drag coefficient

$D_w^i$  = diffusion coefficient of water content

$S_\lambda$  = water generation rate due to cathode side reaction in the catalyst layer  
(Equation 1.20 (p. 236))

$S_{gd}$  = rate of mass change between gas and dissolved phases

$S_{ld}$  = rate of mass change between liquid and dissolved phases

The  $S_{gd}$  and  $S_{ld}$  are expressed as ([10] (p. 355)):

$$S_{gd} = (1-s^\theta) \gamma_{gd} M_{w,H_2O} \frac{\rho_i}{EW} (\lambda_{eq} - \lambda) \quad (1.23)$$

$$S_{ld} = s^\theta \gamma_{ld} M_{w,H_2O} \frac{\rho_i}{EW} (\lambda_{eq} - \lambda) \quad (1.24)$$

where

$\rho_i$  = dry ionomer, or membrane, density

$EW$  = equivalent weight of the membrane

$s$  = liquid saturation

$\lambda_{eq}$  = equilibrium water content

$\gamma_{gd}$  and  $\gamma_{ld}$  = gas and liquid mass exchange rate constants

The  $\gamma_{gd}$ ,  $\gamma_{ld}$  and  $\theta$  are user-specified parameters.

The equilibrium water content is computed as ([10] (p. 355)):

$$\lambda_{eq} = 0.3 + 6a(1 - \tanh(a - 0.5)) + 0.69(\lambda_{a=1} - 3.52)a^{0.5}\left(1 + \tanh\left(\frac{a-0.89}{0.23}\right)\right) + s \cdot (\lambda_{s=1} - \lambda_{a=1}) \quad (1.25)$$

where  $a$  is the water activity defined as:

$$a = p_{wv} / p_{sat} \quad (1.26)$$

where  $p_{wv}$  is the water vapor partial pressure, and  $p_{sat}$  is the saturation pressure.

Both  $\lambda_{s=1}$  and  $\lambda_{a=1}$  in [Equation 1.25 \(p. 237\)](#) are user-specified parameters.

## 1.4.2. The Liquid Phase Model

Liquid water is present in all the porous electrodes and gas channels.

### 1.4.2.1. Liquid Water Transport Equation in the Porous Electrode and the Membrane

The driving force of the liquid water transport is the liquid pressure gradient  $\nabla p_l$  ([\[10\] \(p. 355\)](#)):

$$\frac{\partial}{\partial t}(\varepsilon_i \rho_l s) = \nabla \cdot \left( \frac{\rho_l K K_r}{\mu_l} \nabla p_l \right) + S_{gl} - S_{ld} \quad (1.27)$$

where

$\rho_l$  = liquid water density

$\mu_l$  = liquid dynamic viscosity

$K$  = absolute permeability

$K_r$  = relative permeability

$p_l$  = liquid pressure

$S_{gl}$  = rate of mass change between gas and liquid phases

In the porous gas diffusion and micro-porous layers, the relative permeability is computed as:

$$K_r = s^b \quad (1.28)$$

where  $s$  is liquid saturation, and  $b$  is a user-defined constant.

In the membrane, the relative permeability  $K_r$  is expressed as:

$$K_r = \left( \frac{\frac{M_{w,H_2O}}{\rho_l} \lambda_{s=1} + \frac{EW}{\rho_i}}{\frac{M_{w,H_2O}}{\rho_l} \lambda + \frac{EW}{\rho_i}} \cdot \lambda_{s=1} \right)^2 \quad (1.29)$$

Replacing  $p_l$  in [Equation 1.27 \(p. 238\)](#) with the sum of the capillary pressure  $p_c$  and the gas pressure  $p$ , [Equation 1.27 \(p. 238\)](#) can be rewritten as:

$$\frac{\partial}{\partial t}(\varepsilon_i \rho_l s) = \nabla \cdot \left( \frac{\rho_l K K_r}{\mu_l} \nabla (p_c + p) \right) + S_{gl} - S_{ld} \quad (1.30)$$

The mass transfer rate between the gas and the liquid phases is computed based on the unidirectional diffusion theory [\[2\] \(p. 355\)](#) and [\[10\] \(p. 355\)](#):

$$S_{gl} = \begin{cases} \gamma_{gl} \varepsilon S D_{gl} \frac{M_{w,H_2O}}{RT} p \ln\left(\frac{p - p_{sat}}{p - p_{wv}}\right), & p_{wv} \leq p_{sat} \\ \gamma_{gl} \varepsilon (1-s) D_{gl} \frac{M_{w,H_2O}}{RT} p \ln\left(\frac{p - p_{sat}}{p - p_{wv}}\right), & p_{wv} > p_{sat} \end{cases} \quad (1.31)$$

where  $\varepsilon$  is the porosity,  $\gamma_{gl}$  is the geometric factor of the droplet size (typically on the order of  $10^8 \text{ m}^{-2}$ ), and  $D_{gl}$  has the following form:

$$D_{gl} = \begin{cases} 0.365 \cdot 10^{-4} \left( \frac{T}{343} \right)^{2.334} \cdot \left( \frac{10^5}{p} \right) & \text{cathode} \\ 1.79 \cdot 10^{-4} \left( \frac{T}{343} \right)^{2.334} \cdot \left( \frac{10^5}{p} \right) & \text{anode} \end{cases}$$

The ANSYS Fluent PEMFC module solves for the capillary pressure  $p_c$  (Equation 1.30 (p. 238)). Then, because capillary pressure is a function of saturation, liquid saturation can be computed. Note that even though the capillary pressure is continuous across various porous zones, liquid saturation can be discontinuous at the zone interfaces.

Equation 1.30 (p. 238) is solved in all the regions—from the anode GDL-channel interface all the way to the cathode GDL-channel interface. At these two interface boundaries, liquid water flux is assumed to go out of the GDL and into the gas channel only. No backflow is allowed. The flux is assumed to be driven by the capillary pressure ([10] (p. 355)):

$$f_{liq} = \max[\theta \varepsilon s p_c, 0] \quad (1.32)$$

where  $\theta$  is the coefficient of liquid water removal.

Once the capillary pressure is obtained by solving Equation 1.30 (p. 238), liquid saturation is computed from the following Leverett function:

$$p_c = \begin{cases} \sigma \cos \theta_c \sqrt{\frac{\varepsilon}{K}} J(1-s) & \text{if } \theta_c < 90^\circ \\ \sigma \cos \theta_c \sqrt{\frac{\varepsilon}{K}} J(s) & \text{if } \theta_c > 90^\circ \end{cases} \quad (1.33)$$

$$J(x) = 1.417x - 2.12x^2 + 1.263x^3 \quad (1.34)$$

where  $\sigma$  is the surface tension (N/m), and  $\theta_c$  is the contact angle.

Liquid water will reduce the effective active surface area in the catalyst layers. This is modeled by modifying the transfer currents as follows:

$$R_j = (1-s)^{\gamma_j} R_j \quad (1.35)$$

where  $\gamma_j$  is a user-specified constant.

### 1.4.2.2. Liquid Water Transport Equation in Gas Channels

Liquid water leaves the gas diffusion layers and enters the gas channels. The main purpose of modeling the presence of liquid water in gas channels is to predict the pressure drop increase. In the PEMFC model, liquid water in the channels is tracked using the following correlation:

$$\frac{\partial}{\partial t} (\rho_l s) + \nabla \cdot (\rho_l \vec{v}_l s) = \nabla \cdot (D_{liq} \nabla s) \quad (1.36)$$

where  $D_{liq}$  is the liquid water diffusion coefficient in the gas channel, and  $\vec{v}_l$  is the liquid velocity which is assumed to be a fraction of the gas velocity  $\vec{v}_g$ :

$$\vec{v}_l = \chi \vec{v}_g \quad (1.37)$$

where  $\chi$  is the liquid to gas velocity ratio.

At the anode and cathode flow inlets, liquid saturation  $s = 0$ . The liquid flux calculated from Equation 1.32 (p. 239) is used as a boundary condition at the GDL-channel interfaces for Equation 1.36 (p. 239). Since it is reasonable to assume that the flow is convection-dominated, the phase change in the gas

channel is not considered here. With some meaningful level of saturation in the gas channels, momentum resistance can be constructed to model the pressure drop using the UDF function `resistance_in_channel (real sat)` in `pemfc_user.c`.

## 1.5. Heat Source

Additional volumetric sources in the thermal energy equation are present because not all chemical energy released in the electrochemical reactions can be converted to electrical work due to phase changes and irreversibilities of the processes. Volumetric heat source terms in various zones are listed in [Table 1.1: Volumetric Heat Source Terms \(p. 240\) \(\[10\] \(p. 355\)\)](#).

**Table 1.1: Volumetric Heat Source Terms**

Zone	Additional Source Term
GDL+MPL	$i_s^2 / \sigma_{sol} - S_{gl} \cdot L$
Anode Catalyst Layer	$R_{an} \left( \eta_{an} - \frac{T \Delta S_{an}}{2F} \right) + \frac{i_s^2}{\sigma_{sol}} + \frac{i_m^2}{\sigma_{mem}} - (S_{dl} + S_{gl}) \cdot L$
Cathode Catalyst Layer	$R_{cat} \left( -\eta_{cat} - \frac{T \Delta S_{cat}}{2F} \right) + \frac{i_s^2}{\sigma_{sol}} + \frac{i_m^2}{\sigma_{mem}} - (S_{dl} + S_{gl}) \cdot L$
Membrane (solid)	$i_m^2 / \sigma_{mem}$
Current Collector (solid)	$i_s^2 / \sigma_{sol}$
Gas Channels	None

In this table,  $i_s$  and  $i_m$  are the magnitude of the solid phase and membrane phase current density, respectively, and  $L < 0$  is the latent heat due to water condensation.

## 1.6. Properties

- Gas Phase Species Diffusivity

Gas phase species diffusivities can be computed either by using the dilute approximation method or by using the full multicomponent method. With the dilute approximation method, we have

$$D_i = \varepsilon^{1.5} (1-s)^{r_s} D_i^0 \left( \frac{p_0}{p} \right)^{\gamma_p} \left( \frac{T}{T_0} \right)^{\gamma_t} \quad (1.38)$$

where  $\varepsilon$  is the porosity of the porous medium,  $D_i^0$  is the mass diffusivity of species  $i$  at reference temperature and pressure ( $P_0, T_0$ ) [12] (p. 355). These reference values and the exponents ( $\gamma_p, \gamma_t$ ) as well as the exponent of pore blockage ( $r_s$ ) are defined in the PEMFC user-defined functions (UDF) as,

$$\begin{aligned} p_0 &= 101325 N/m^2 \\ T_0 &= 300 K \\ \gamma_p &= 1.0 \\ \gamma_t &= 1.5 \\ r_s &= 2.5 \end{aligned} \quad (1.39)$$

In addition to [Equation 1.38 \(p. 240\)](#), the ANSYS Fluent PEMFC model also contains a method to compute the gas phase species diffusion (a full multicomponent diffusion method with corrections to account for the porous media tortuosity):

$$D_{eff}^{ij} = (1-s)^{r_s} \varepsilon^{1.5} D^{ij} \quad (1.40)$$

where  $D_{eff}^{ij}$  is the effective gas species diffusivity, and  $D^{ij}$  is the gas species mass diffusivity computed by the full multicomponent diffusion method (as described in [Full Multicomponent Diffusion](#) in the separate ANSYS Fluent [User's Guide](#)). Note that  $\varepsilon^{1.5}$  in [Equation 1.40 \(p. 241\)](#) is used to model the effect of tortuosity. While this is implemented as the default method in the PEMFC, you can overwrite it with your own correction methods by using the user-modifiable routines that are provided.

Properties such as electrolyte phase electrical conductivity, water diffusivity, and the osmotic drag coefficient are evaluated as functions of the water content, using various correlations as suggested by [\[11\] \(p. 355\)](#). To capture the relevant physics of the problem, various properties of the membrane are incorporated into the model as default options. You can, however, directly incorporate your own formulations and data for these properties by editing the functions defined in the provided source code file called `pem_user.c` and compiling the code yourself. For more information, see [User-Accessible Functions \(p. 272\)](#).

- [Electrolyte Phase \(Ionic\) Conductivity](#)

The electrolyte (also called the membrane) phase conductivity is modeled based upon [\[11\] \(p. 355\)](#):

$$\sigma_{mem} = \Gamma_i (0.514\lambda - 0.326)^{\omega_i} e^{E_i(\frac{1}{303} - \frac{1}{T})} \quad (1.41)$$

where  $\lambda$  is the water content, and  $E_i$  is the activation energy for the temperature correction term.

The  $\Gamma_i$  is calculated as ([\[10\] \(p. 355\)](#)):

$$\Gamma_i = \begin{cases} \beta_{mem} & \text{in membrane} \\ \beta_{an} \frac{\varsigma_a}{\tau_a} & \text{in anode catalyst} \\ \beta_{ca} \frac{\varsigma_c}{\tau_c} & \text{in cathode catalyst} \end{cases} \quad (1.42)$$

where

$\varsigma_a$  = anode ionomer volume fraction

$\varsigma_c$  = cathode ionomer volume fraction

$\tau_a$  = anode ionomer tortuosity

$\tau_c$  = cathode ionomer tortuosity

Two model constants,  $\beta$  and  $\omega$  are introduced in ANSYS Fluent for generality. [Equation 1.41 \(p. 241\)](#) becomes the original correlation from [\[11\] \(p. 355\)](#) when  $\beta=\omega=1$ .

- [Diffusivity of Water Content](#)

The diffusivity coefficient in the water content transport equation is obtained from [\[13\] \(p. 355\)](#):

$$D_w^i = \eta_\lambda \frac{\rho_i}{EW} f(\lambda) \quad (1.43)$$

where  $\eta_\lambda$  is a user-specified coefficient for generality, and the function  $f(\lambda)$  has the following form:

$$f(\lambda) = 4.1e^{-10} \left( \frac{\lambda}{25} \right)^{0.15} \left[ 1 + \tanh \left( \frac{\lambda - 2.5}{1.4} \right) \right]$$

- Osmotic Drag Coefficient

$$n_d = \eta_{osm} g(\lambda) \quad (1.44)$$

where  $\eta_{osm}$  is a user-specified coefficient for generality with the default value of 1.0. The default formulation for  $g(\lambda)$  is:

$$g(\lambda) = 2.5 \frac{\lambda}{22}$$

This formulation can be changed using the provided user-accessible function code file `pemfc_user.c`.

- Saturation Pressure

The saturation pressure is calculated, in terms of *atm*, as,

$$\begin{aligned} \log_{10} P_{sat} = & -2.1794 + 0.02953(T - 273.17) \\ & - 9.1837 \times 10^{-5}(T - 273.17)^2 \\ & + 1.4454 \times 10^{-7}(T - 273.17)^3 \end{aligned} \quad (1.45)$$

ANSYS Fluent allows you to provide a custom formulation by modifying the user-accessible function code file `pemfc_user.c`.

## 1.7. Transient Simulations

Dynamics response to changes in operating conditions as a function of time can be modeled using the PEM Fuel Cell module. Examples include modeling a change in the cell voltage or current density, or inlet mass flow rates at the anode and/or the cathode. The procedure for setting up and solving transient PEM Fuel Cell problems is the same as that used for a normal ANSYS Fluent transient problem as discussed in the ANSYS Fluent [User's Guide](#).

Assuming that the time scales associated with the electric fields are much smaller than those associated with the flow and thermal fields, the steady-state equations are retained for the two electric potentials ([Equation 1.3 \(p. 232\)](#) and [Equation 1.4 \(p. 232\)](#)). Transient terms in all other equations such as momentum transport, energy transport, species transport, dissolved water transport ([Equation 1.22 \(p. 237\)](#)), liquid water transport (the capillary pressure [Equation 1.30 \(p. 238\)](#)), and the liquid saturation in gas channels ([Equation 1.36 \(p. 239\)](#)), are activated.

## 1.8. Leakage Current (Cross-Over Current)

The effect of hydrogen cross-over from anode to cathode through the membrane is modeled by a total leakage current  $I_l$  (A). In addition to the source terms expressed by [Equation 1.18 \(p. 236\)](#) through [Equation 1.20 \(p. 236\)](#), the following extra volumetric source terms are added to the corresponding equations:

$$S_{H_2} = -\frac{M_{w,H_2}}{2F} \cdot \frac{I_l}{Vol_{anode}} \quad (1.46)$$

$$S_{O_2} = -\frac{M_{w,O_2}}{4F} \cdot \frac{I_l}{Vol_{cathode}} \quad (1.47)$$

$$S_\lambda = \frac{M_{w,H_2O}}{2F} \cdot \frac{I_l}{Vol_{cathode}} \quad (1.48)$$

Here,  $Vol_{cathode}$  and  $Vol_{anode}$  are the volumes of the catalytic layers on the anode and cathode sides, and 2 and 4 are the numbers of electrons per mole of reactants and products.

Accordingly, the volumetric leakage current ( $I_l/Vol_{cathode}$ ) is subtracted from the cathode transfer current ( $R_{cat}$ ) that is computed by [Equation 1.6 \(p. 234\)](#) and [Equation 1.8 \(p. 234\)](#).



---

## Chapter 2: Using the PEMFC Model

---

The procedure for setting up and solving fuel cell problems using the PEMFC model is described in detail in this chapter. Refer to the following sections for more information:

- 2.1. Overview and Limitations
- 2.2. Geometry Definition for the PEMFC Model
- 2.3. Installing the PEMFC Model
- 2.4. Loading the PEMFC Module
- 2.5. Setting Up the PEMFC Module
- 2.6. Modeling PEM Fuel Cells
- 2.7. PEMFC Model Boundary Conditions
- 2.8. Solution Guidelines for the PEMFC Model
- 2.9. Postprocessing the PEMFC Model
- 2.10. User-Accessible Functions
- 2.11. Using the PEMFC Text User Interface

### 2.1. Overview and Limitations

The ANSYS Fluent PEMFC model comprises several user-defined functions (UDFs) and a graphical user interface. The potential fields are solved as user-defined scalars. The capillary pressure, the water content (water in dissolved phase), and liquid saturation in the gas channels are also solved as user-defined scalars. The electrochemical reactions occurring in the catalyst layers are modeled through various source terms while other model parameters are handled through the user interface. The PEMFC model can be used in parallel ANSYS Fluent as well.

Note the following limitation when using the PEMFC model:

- The anisotropic species diffusivity option is not compatible with the PEMFC model.

### 2.2. Geometry Definition for the PEMFC Model

Due to the fact that there are a number of different physical zones associated with the fuel cell, the following regions must be present in the fuel cell mesh (see [Figure 1.1: Schematic of a PEM Fuel Cell \(p. 231\)](#)):

- Anode flow channel
- Anode gas diffusion layer
- Anode micro-porous layer (optional)
- Anode catalyst layer
- Membrane layer (solid)
- Cathode catalyst layer
- Cathode micro-porous layer (optional)

- Cathode gas diffusion layer
- Cathode flow channel

The following zones have to be identified, if present in the fuel cell mesh:

- Anode current collector (solid)
- Cathode current collector (solid)
- Coolant channel

## 2.3. Installing the PEMFC Model

The PEMFC model is provided as an add-on module with the standard ANSYS Fluent licensed software. The module is installed with the standard installation of ANSYS Fluent in a directory called addons/pemfc in your installation area. The PEMFC module consists of a UDF library and a pre-compiled scheme library, which must be loaded and activated before calculations can be performed.

## 2.4. Loading the PEMFC Module

The PEMFC module is loaded into ANSYS Fluent through the text user interface (TUI). The module can only be loaded after a valid ANSYS Fluent case file has been set or read. The text command to load the add-on module is

```
define → models → addon-module
```

A list of ANSYS Fluent add-on modules is displayed:

```
> /define/models/addon-module
Fluent Addon Modules:
  0. None
  1. MHD Model
  2. Fiber Model
  3. Fuel Cell and Electrolysis Model
  4. SOFC Model with Unresolved Electrolyte
  5. Population Balance Model
  6. Adjoint Solver
  7. Battery Module
  8. MSMD Battery Model
  9. PEM Fuel Cell Model
Enter Module Number: [0] 9
```

Select the PEM Fuel Cell Model by entering the module number 9. During the loading process, a scheme library (containing the graphical and text user interface) and a UDF library (containing a set of user defined functions) are loaded into ANSYS Fluent.

## 2.5. Setting Up the PEMFC Module

The PEMFC model can be used to model polymer electrolyte membrane fuel cells (PEMFC). The following describes an overview of the procedure required in order to use the PEMFC model in ANSYS Fluent.

1. Start ANSYS Fluent.

You must start ANSYS Fluent in 3D double-precision mode. Note that the PEMFC model is only available in 3D.

2. Read the case or mesh file.

3. Scale the mesh, if necessary.
4. Use the **PEM Fuel Cell Model** dialog box to define the fuel cell model parameters.
5. Define material properties.
6. Set the operating conditions.
7. Set the boundary conditions.
8. Start the calculations.
9. Save the case and data files.
10. Process your results.

### **Important**

The **PEM Fuel Cell Model** dialog box greatly simplifies the input of parameters and boundary conditions, but it does not replace the boundary conditions interface. Therefore, it is a good policy to start the setup with the **PEM Fuel Cell Model** dialog box and do the finishing steps for boundary conditions afterwards.

### **Important**

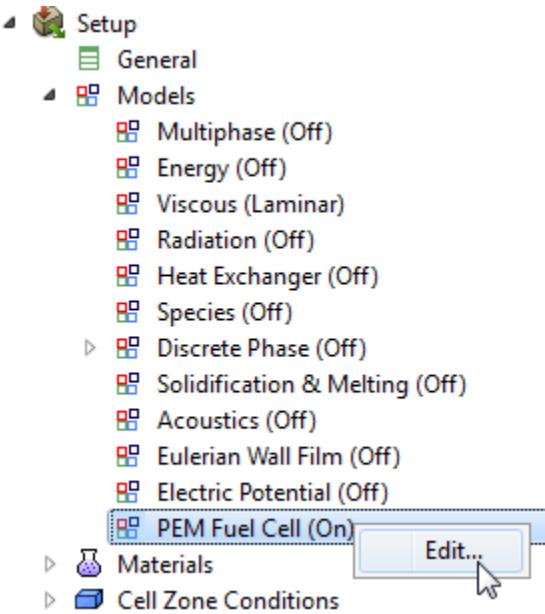
Note that the majority of this chapter describes how to set up the ANSYS Fluent PEMFC model using the graphical user interface. You can also perform various tasks using the text user interface. For more information, see [Using the PEMFC Text User Interface \(p. 277\)](#).

## **2.6. Modeling PEM Fuel Cells**

Once the module has been loaded, in order to set fuel cell model parameters and assign properties to the relevant regions in your fuel cell, you need to access the fuel cell graphical user interface (the **PEM Fuel Cell Model** dialog box).

To open the **PEM Fuel Cell Model** dialog box: In the tree, under the **Models** branch, right-click **PEM Fuel Cel** and select **Edit...** in the menu that opens.



**Figure 2.1: The PEMFC Option in the Tree**

By default, the PEMFC model is enabled.

Using the **PEM Fuel Cell Model** dialog box, you can identify the relevant zones for the current collectors, flow channels, gas diffusion layers, micro porous layers, catalyst layers, and the electrolyte (membrane). You can specify the following inputs using the **PEM Fuel Cell Model** dialog box. Optional inputs are indicated as such.

1. Set the appropriate options for the fuel cell model.
2. Set the various parameters for the fuel cell model.
3. Select the appropriate zones and specify the properties on the anode side.
4. Select the appropriate zones and specify the properties of the electrolyte/membrane.
5. Select the appropriate zones and specify the properties on the cathode side.
6. Provide input for advanced features such as contact resistivities, coolant channel properties, or stack management settings (optional).
7. Set solution controls such as under-relaxation factors (optional).
8. Provide input to assist reporting (optional).

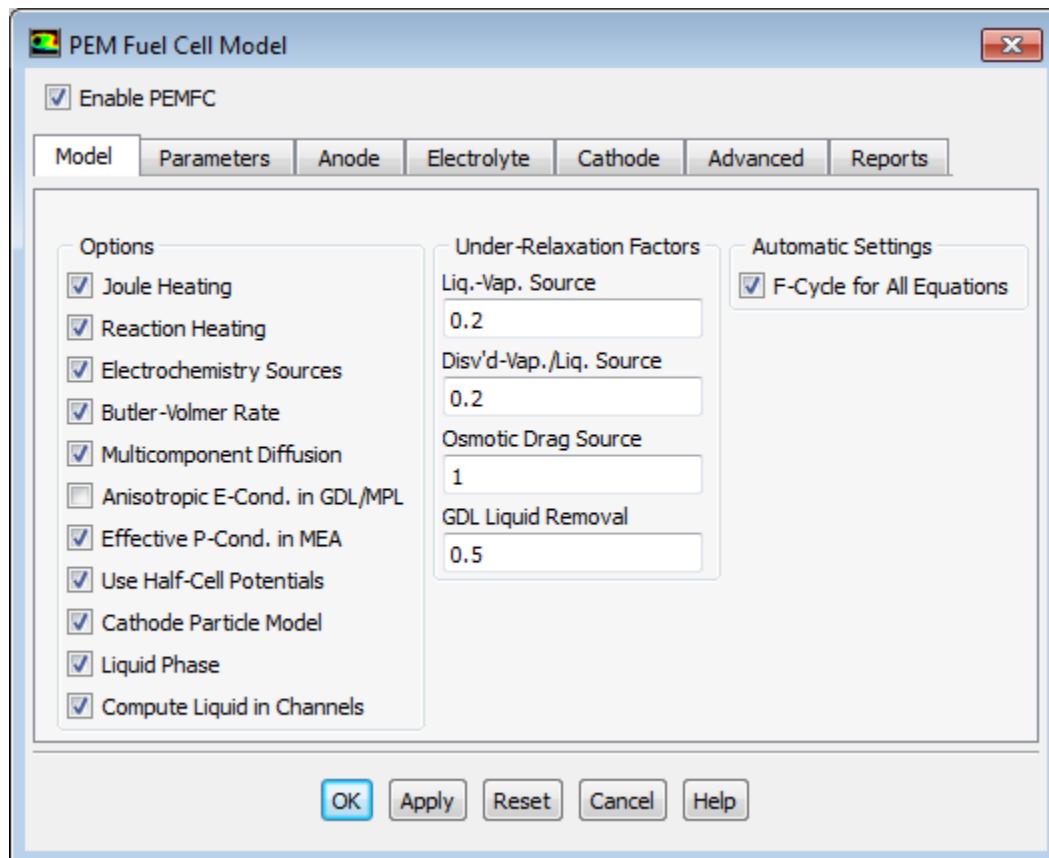
Refer to the following sections for more information:

- [2.6.1. Specifying Model Options](#)
- [2.6.2. Specifying Model Parameters](#)
- [2.6.3. Specifying Anode Properties](#)
- [2.6.4. Specifying Electrolyte/Membrane Properties](#)
- [2.6.5. Specifying Cathode Properties](#)
- [2.6.6. Setting Advanced Properties](#)
- [2.6.7. Reporting on the Solution](#)

## 2.6.1. Specifying Model Options

The **Model** tab of the **PEM Fuel Cell Model** dialog box allows you to select or cancel the selection of various options when solving a fuel cell problem.

**Figure 2.2: The Model Options in the PEM Fuel Cell Model Dialog Box**



### Options

Several fuel cell model options are available including:

#### Joule Heating

takes into account ohmic heating. This option includes the  $i^2/\sigma$  terms in the energy source term in the calculations. See [Table 1.1: Volumetric Heat Source Terms \(p. 240\)](#) for the list of additional volumetric sources in the thermal energy equation.

#### Reaction Heating

takes into account the heat generated by the electrochemical reactions.

#### Electrochemistry Sources

allows the PEMFC model to take electrochemistry effects into account. If you are only interested in the basic flow field throughout the fuel cell, you can turn off the **Electrochemistry Sources** option in order to suppress most effects of the PEMFC model. You may also turn off the effect of these sources in order to obtain a fluid flow initially, and then turn it back on.

**Butler-Volmer Rate**

(the default) is used to compute the transfer currents inside the catalyst layers. If this option is turned off, the Tafel approximation ([Equation 1.7 \(p. 234\)](#) and [Equation 1.8 \(p. 234\)](#)) is used.

**Liquid Phase**

takes into account liquid phase calculations. Use this option if you are solving for liquid transport in the fuel cell.

**Multicomponent Diffusion**

is used to compute the gas species mass diffusivity using the full multicomponent diffusion method as described in [Equation 1.40 \(p. 241\)](#), as opposed to the default option that uses [Equation 1.38 \(p. 240\)](#).

**Anisotropic E-Cond. in Porous Electrode**

is used to model the typically non-isotropic electrical conductivity. It is applicable only for porous electrodes (gas diffusion layers and micro porous layers).

Due to the fibrous structure of the porous material that is used for the electrodes (or gas diffusion layer), the electrical conductivity is typically non-isotropical, with the cross-plane components being orders of magnitude smaller than the in-plane components. This can be modeled using the **Anisotropic E-Conductivity in Porous Electrode** setting. When this option is enabled, the **Electrical Conductivity** for the solid material used in the porous electrodes is no longer used. Instead, you need to specify, for this solid material, the electrical conductivity by choosing one of the three non-isotropical options for the UDS diffusivity (UDS-0). The three options are: **anisotropic**; **orthotropic**; and **cyl-orthotropic**. For more information about these UDS Diffusivity options, refer to the ANSYS Fluent User's Guide.

For example, to use this feature, perform the following steps:

- Select the **Anisotropic E-Conductivity in Porous Electrode** option in the **Model** tab of the **PEM Fuel Cell Model** dialog box.
- In the **Create/Edit Materials** dialog box, select **defined-per-uds** for **UDS Diffusivity** for the solid material that is to be used for the porous electrode.
- Select one of the three options for UDS-0: **anisotropic**; **orthotropic**; or **cyl-orthotropic** and set the appropriate values.

---

**Important**

Note that, in this case, the **Electrical Conductivity** for *this* solid material is ignored.

---

**Effective P-Cond. in MEA**

enables the computation of the electrolyte phase conductivity  $\sigma_{mem}$  with consideration of ionomer volume fraction and tortuosity in the catalyst layers using [Equation 1.41 \(p. 241\)](#). If this option is disabled, then  $\varsigma_a/\tau_a = \varsigma_c/\tau_c = 1$  and  $\beta_{an} = \beta_{ca} = \beta_{mem}$ .

**Use Half-Cell Potentials**

enables the calculations of the anode and cathode half-cell potentials  $U_{an}^0$  and  $U_{cat}^0$  using the Nernst equations ([Equation 1.13 \(p. 235\)](#) and [Equation 1.14 \(p. 235\)](#)). If this option is disabled, the constant open-circuit voltage is assumed, and the  $U_{an}^0$  and  $U_{cat}^0$  are set to zero.

**Cathode Particle Model**

enables the cathode particle model. For information about this model, see [The Cathode Particle Model \(p. 235\)](#).

**Liquid Phase**

enables the liquid water transport equations in the computations. By default, this option is enabled.

**Compute Liquid in Channels**

enables the solution of the liquid saturation equation in the gas channels. By default, this option is disabled, and the equation is not solved.

***Under-Relaxation Factors***

You can use the **Under-Relaxation Factors** fields to influence the solution process.

**Liquid-Vapor**

The source term  $S_{gl}$  in [Equation 1.30 \(p. 238\)](#) usually requires under-relaxation. Since the  $S_{gl}$  is computed explicitly as a source term for the capillary pressure equation ([Equation 1.30 \(p. 238\)](#)), the under-relaxation factor usually needs to be a small value in order to obtain convergence. You can change the default value for the under-relaxation factor by changing the value for **Liquid-Vapor**.

**Disv'd-Vapor/Liquid**

The source terms  $S_{ld}$  and  $S_{gd}$  in [Equation 1.22 \(p. 237\)](#) usually require under-relaxation also. You can change the default value for the under-relaxation factor by changing the value for **Disv'd-Vapor/Liquid**.

**Osmotic Drag Source**

You can also specify an under-relaxation factor for the osmotic drag term, namely the left-hand side term of [Equation 1.22 \(p. 237\)](#) by changing the value for **Osmotic Drag Source**.

**GDL Liquid Removal**

The liquid water flux that goes out of the gas diffusion layer (GDL) at the GDL-Channel interface ([Equation 1.32 \(p. 239\)](#)) can also be under-relaxed using **GDL Liquid Removal**.

***Automatic Settings***

The following parallel multigrid solver control parameter is available:

**F-cycle for All Equations**

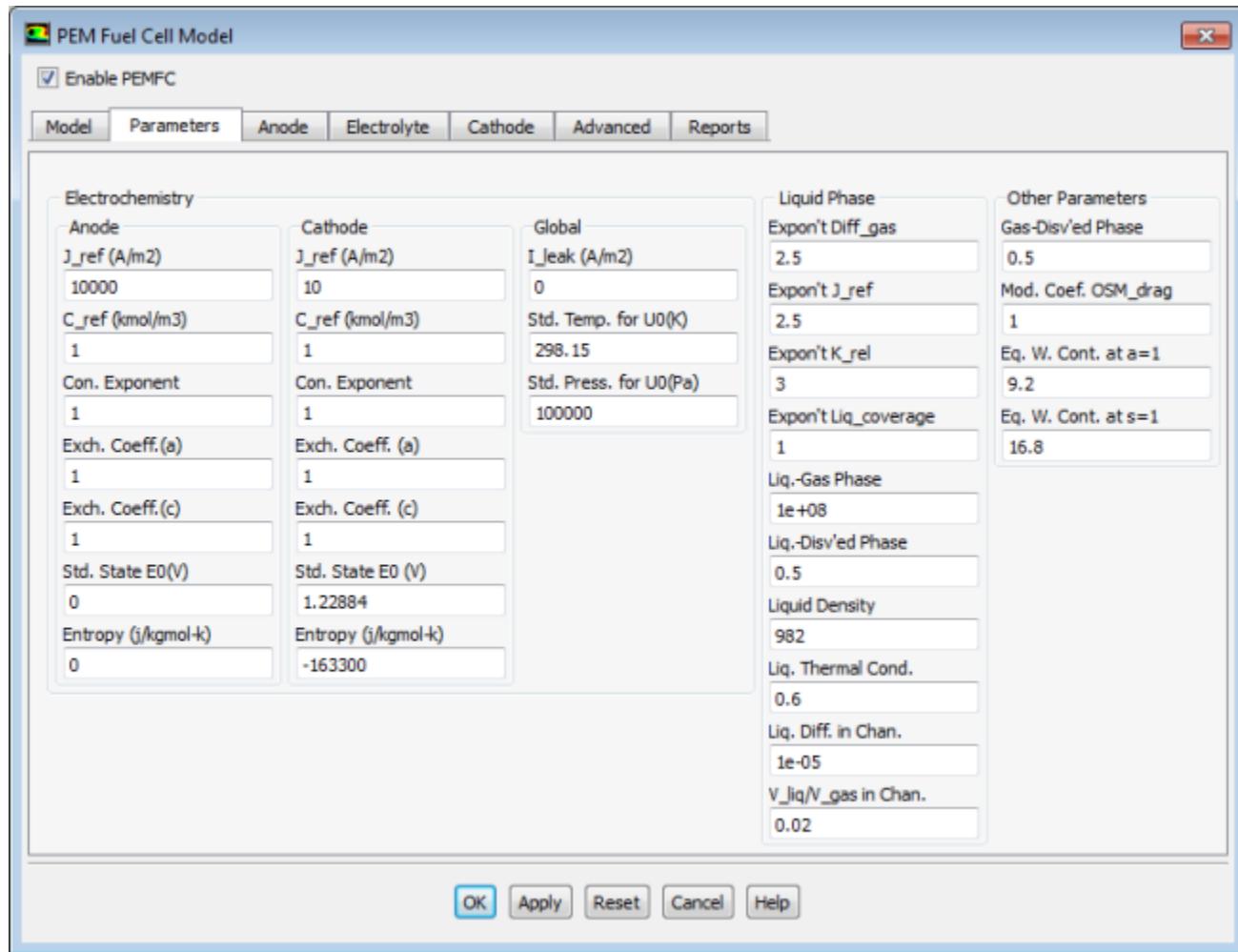
sets the multigrid cycle type to F cycle for all equations that are being solved. This control overrides the equation cycle settings in the **Advanced Solution Controls** dialog box.

**Note**

Using F-cycle for PEMFC computations is the best practice approach.

***2.6.2. Specifying Model Parameters***

You can use the **Parameters** tab of the **PEM Fuel Cell Model** dialog box to specify the electrochemistry parameters for the PEMFC model, reference diffusivities for the reactants and other model parameters.

**Figure 2.3: The Parameters Tab of the PEM Fuel Cell Model Dialog Box**

## Electrochemistry

There are various parameters under **Electrochemistry** in the **PEM Fuel Cell Model** dialog box. For both the anode and the cathode, you can set the following parameters or leave the default values:

### **J\_ref**

corresponds to  $j_{an}^{ref}$  and  $j_{cat}^{ref}$ , the reference exchange current density from [Equation 1.9 \(p. 235\)](#) and [Equation 1.10 \(p. 235\)](#).

### **C\_ref**

corresponds to the reference concentration ( $[H_2]_{ref}$  and  $[O_2]_{ref}$ ) with units of 1 kgmol/m<sup>3</sup> (see [Equation 1.5 \(p. 234\)](#) and [Equation 1.6 \(p. 234\)](#)).

### **Con. Exponent**

corresponds to  $\gamma$ , the concentration dependence from [Equation 1.5 \(p. 234\)](#).

### **Exch. Coeff. (a), Exch. Coeff. (c)**

are the transfer coefficients  $\alpha_{an}$  and  $\alpha_{cat}$  from [Equation 1.5 \(p. 234\)](#) and [Equation 1.6 \(p. 234\)](#), respectively.

**Std. State E0**

are the reversible potentials  $E_{an}^0$  and  $E_{cat}^0$  in [Equation 1.13 \(p. 235\)](#) and [Equation 1.14 \(p. 235\)](#), respectively. This field appears only when **Use Half-Cell Potentials** option is selected under the **Model** tab.

**Entropy**

is the reaction entropy  $\Delta S_{an}$  and  $\Delta S_{cat}$  in [Equation 1.13 \(p. 235\)](#) and [Equation 1.14 \(p. 235\)](#).

**V\_Open**

corresponds to the constant value assigned to cathode half-cell potential  $U_{an}^0$  and  $U_{cat}^0$ . This parameter appears only when **Use Half-Cell Potentials** option is not selected under the **Model** tab.

**I\_leak**

is the leakage current density ( $A/m^2$ ). It is used to compute the total leakage current  $I_l$  (in [Equation 1.46 \(p. 242\)](#) through [Equation 1.48 \(p. 243\)](#)) by:

$$I_l = I_{leak} A_e$$

where  $A_e$  is the electrolyte projected area (specified under the **Reports** tab). When leakage through the electrolyte occurs, the fuel cell generates less current especially for cases with low values of fuel or air utilization. Note that you can also specify the total leakage current through the user-defined function `Leakge_Current()`. For more information, see [User-Accessible Functions \(p. 272\)](#).

**Std. Temp. for U0 (K)**

is the reference standard state temperature which is used for half-cell potentials computations in the Nernst equations ([Equation 1.13 \(p. 235\)](#) and [Equation 1.14 \(p. 235\)](#)). This parameter appears only when the **Use Half-Cell Potentials** option is selected under the **Model** tab.

**Std. Temp. for U0 (Pa)**

is the reference standard state pressure which is used for half-cell potentials computations in the Nernst equations ([Equation 1.13 \(p. 235\)](#) and [Equation 1.14 \(p. 235\)](#)). This parameter appears only when the **Use Half-Cell Potentials** option is selected under the **Model** tab.

**Note**

The default values of the model parameters are determined based on various data available in the literature ([8] (p. 355), [9] (p. 355), [12] (p. 355) [13] (p. 355), and others).

**Reference Diffusivity**

The parameters in the **Reference Diffusivity** group box appear only if the **Multicomponent Diffusion** option is turned off in the **Model** tab. These parameters correspond to the species mass diffusivity  $D_i^0$  from [Equation 1.38 \(p. 240\)](#).

**Liquid Phase**

When the **Liquid Phase** option is selected under the **Model** tab, you can specify the following parameters that appear in the **Liquid Phase** group box:

**Expon't Diff\_gas**

corresponds to  $r_s$  from [Equation 1.38 \(p. 240\)](#) for multiphase PEMFC calculations.

**Expon't J\_ref**

is the exponent  $\gamma_j$  that is used to modify  $R_{an}$  and  $R_{ca}$  according to Equation 1.35 (p. 239) to account for liquid blockage to the reaction surface in Equation 1.5 (p. 234) through Equation 1.8 (p. 234).

**Expon't K\_rel**

is the exponent  $b$  that is used to compute the relative permeability in  $K_r$  (Equation 1.28 (p. 238)).

**Expon't Liq\_coverage**

is the exponent  $\theta$  that is used to compute the phase-change rates  $S_{gd}$  and  $S_{ld}$  (Equation 1.23 (p. 237) and Equation 1.24 (p. 237)).

**Liq.-Gas Phase**

is the geometric factor of the droplet size  $\gamma_{gl}$  that is used to compute the rate of mass change between gas and liquid phases  $S_{gl}$  (Equation 1.31 (p. 238)).

**Liq.-Disv'ed Phase**

is the mass exchange rate constant  $\gamma_{ld}$  that is used to compute the rate of mass change between liquid and dissolved phases  $S_{ld}$  (Equation 1.24 (p. 237)).

**Liquid Density, Liq. Thermal Cond.**

are the density  $\rho_l$  and thermal conductivity  $k_l$  of liquid water, respectively. Note that the effective thermal conductivity in porous media is computed as:

$$k_{eff} = sk_l + (1-s)[\varepsilon k_{gas} + (1-\varepsilon)k_{solid}] \quad (2.1)$$

**Liq. Diff. in Chan.**

is the  $D_{liq}$  in Equation 1.36 (p. 239). This parameter appears only when the **Compute Liquid In Channel** option is selected under the **Model** tab.

**V\_liq/V\_gas in Chan.**

is the liquid to gas velocity ratio in the channel ( $\chi$  in Equation 1.36 (p. 239)). This parameter appears only when the **Compute Liquid In Channel** option is selected under the **Model** tab.

## Other Parameters

Under **Other Parameters**, you can specify the following:

**Gas-Disv'ed Phase**

is the mass exchange rate constant  $\gamma_{gd}$  that is used to compute the rate of mass change between gas and dissolved phases  $S_{gd}$  (Equation 1.23 (p. 237)).

**Mod. Coef. OSM\_drag**

is the constant  $\eta_{osm}$  with the default value of 1.0 that is used to generalize the default osmotic drag coefficient  $n_d$  (Equation 1.44 (p. 242)).

**Eq. W. Cont. at a=1**

is the equilibrium water content at the water activity of 1,  $\lambda_{a=1}$ , that is used in computing the equilibrium water content  $\lambda_{eq}$  (Equation 1.25 (p. 237)).

**Eq.W. Cont. at s=1**

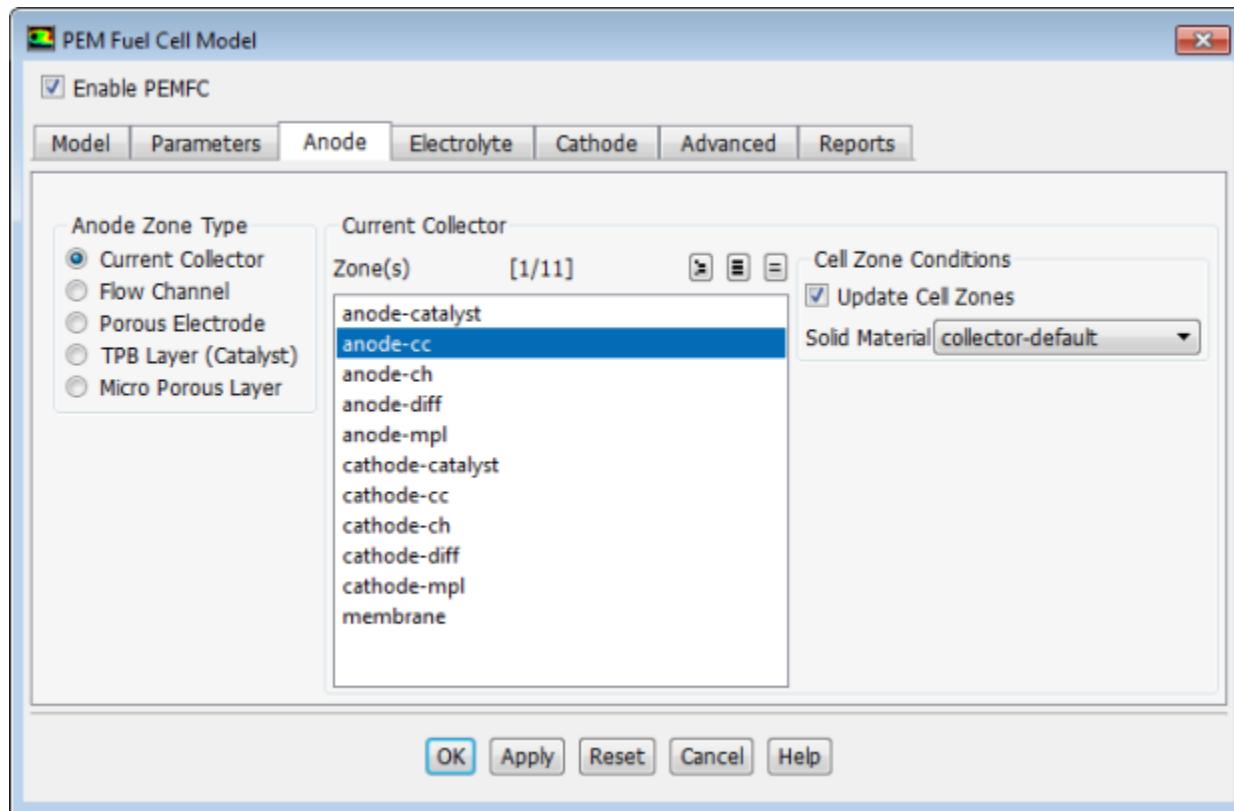
is the equilibrium water content at water saturation of 1,  $\lambda_{s=1}$ , that is used in computing the equilibrium water content  $\lambda_{eq}$  (Equation 1.25 (p. 237)). This item appears only if the **Liquid Phase** option is selected under the **Model** tab.

## 2.6.3. Specifying Anode Properties

You can use the **Anode** tab of the **PEM Fuel Cell Model** dialog box to specify zones and properties of the current collector, the flow channel, the diffusion layer, the micro porous layer (optional), and the catalyst layer for the anode portion of the fuel cell.

### 2.6.3.1. Specifying Current Collector Properties for the Anode

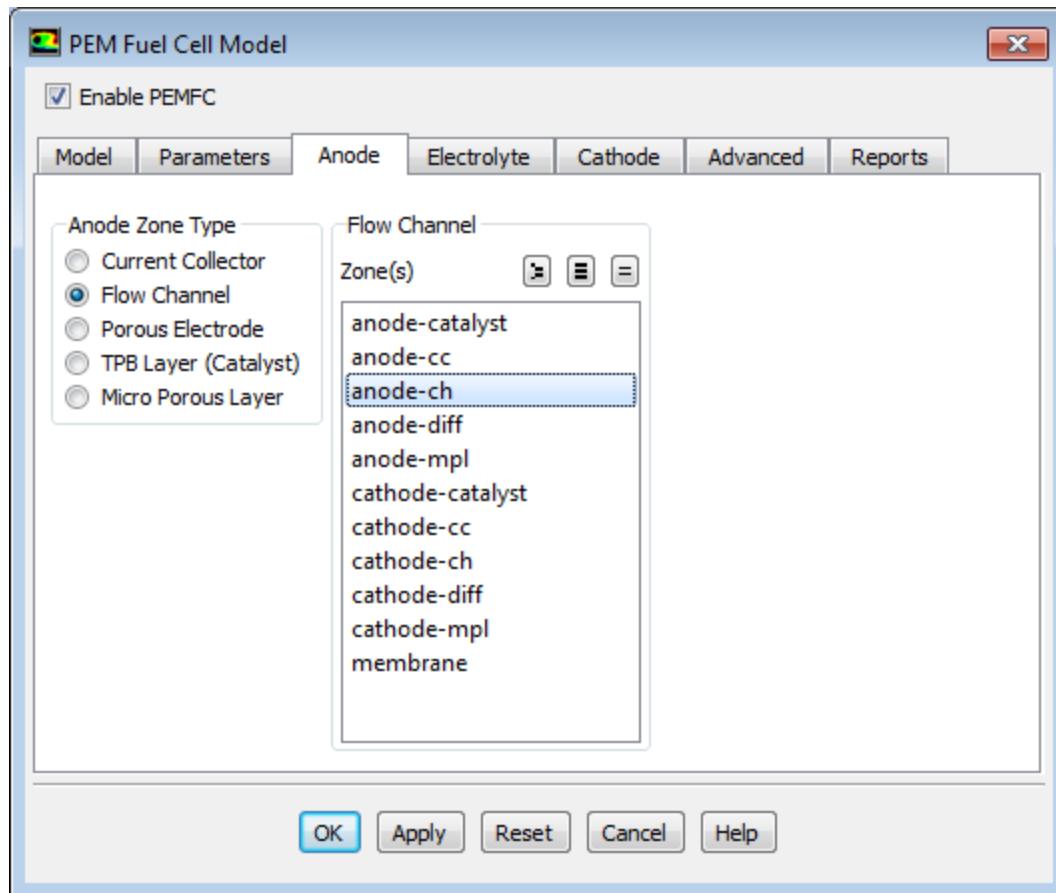
**Figure 2.4: The Anode Tab of the PEM Fuel Cell Model Dialog Box with Current Collector Selected**



- Under **Anode Zone Type**, select **Current Collector**.
- From the **Zone(s)** selection list, select a corresponding zone. If you are modeling a fuel cell stack, then you must select *all* zones of a particular type as a group.
- From the **Solid Material** drop-down list, select the appropriate material. You can use the **Create/Edit Materials** dialog box to customize solid materials. Note that for the **Electrical Conductivity**, you can only select a constant value in the **Create/Edit Materials** dialog box. The solid electrical conductivity value is the diffusivity of the solid phase potential in the solid zones.

### 2.6.3.2. Specifying Flow Channel Properties for the Anode

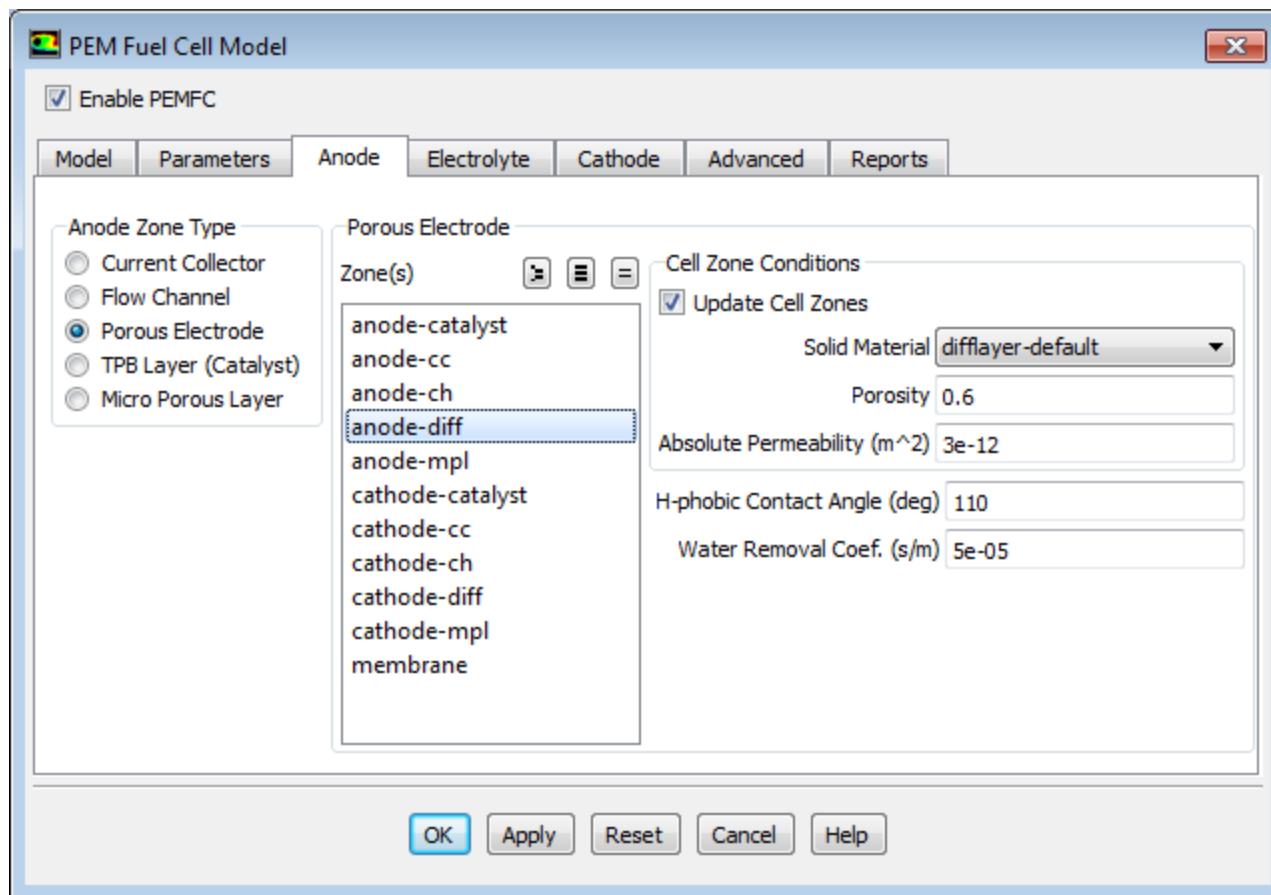
Figure 2.5: The Anode Tab of the PEM Fuel Cell Model Dialog Box with Flow Channel Selected



1. Under **Anode Zone Type**, select **Flow Channel**.
2. From the **Zone(s)** selection list, select an appropriate zone.

### 2.6.3.3. Specifying Porous Electrode Properties for the Anode

Figure 2.6: The Anode Tab of the PEM Fuel Cell Model Dialog Box with Porous Electrode Selected



- Under **Anode Zone Type**, select **Porous Electrode**.
- From the **Zone(s)** selection list, select a corresponding zone. If you are modeling a fuel cell stack, then you must select *all* zones of a particular type as a group.
- From the **Solid Material** drop-down list, select the appropriate material. You can use the **Create/Edit Materials** dialog box to customize solid materials. Note that for the **Electrical Conductivity**, you can only select a constant value in the **Create/Edit Materials** dialog box. The solid electrical conductivity value is the diffusivity of the solid phase potential in the solid zones.
- Specify values for the following parameters:

#### Porosity

is  $\varepsilon$  in [Equation 1.38 \(p. 240\)](#).

#### Absolute Permeability

is  $K$  in [Equation 1.27 \(p. 238\)](#).

#### H-phobic Contact Angle

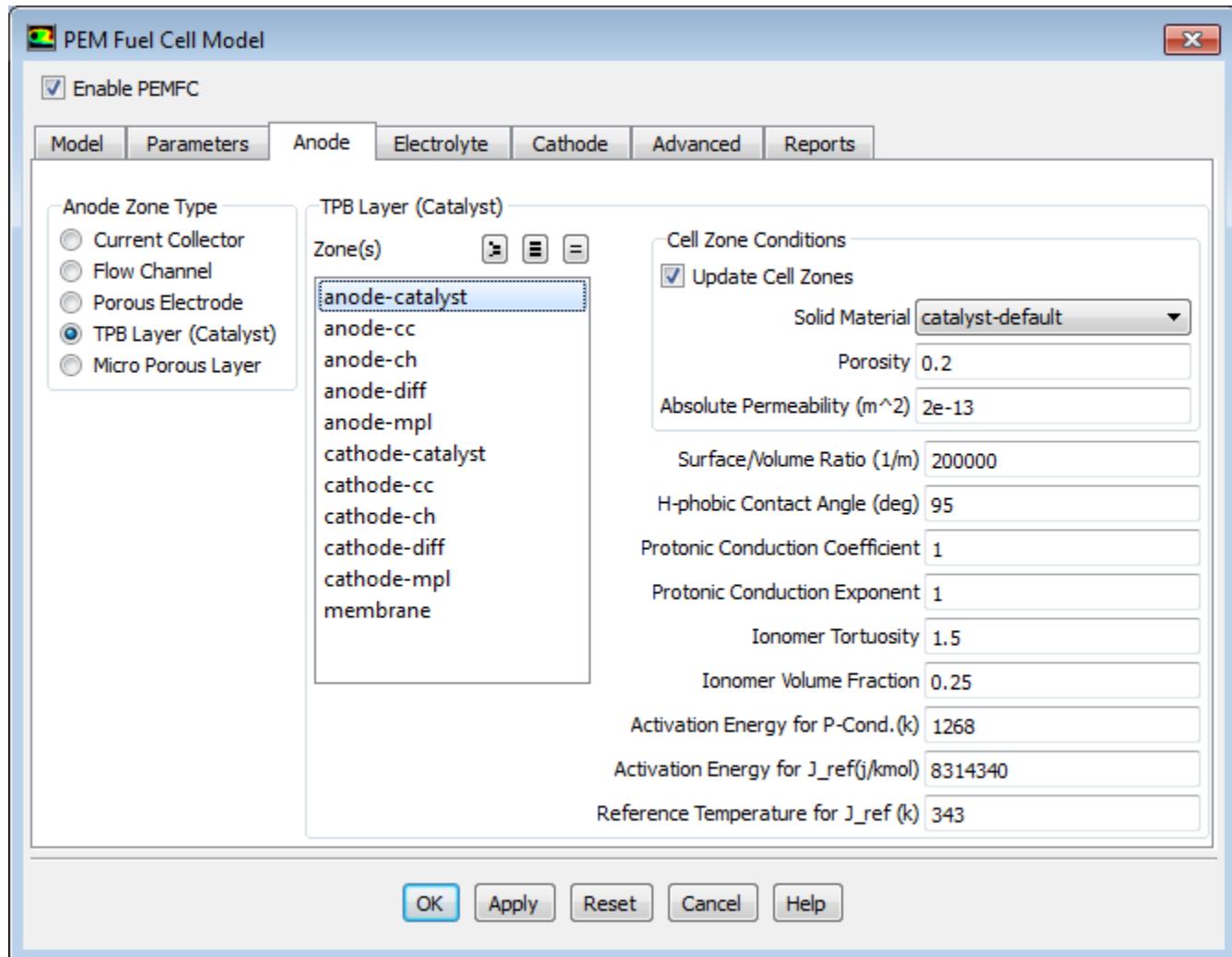
hydrophobic contact angle  $\theta_c$  in [Equation 1.33 \(p. 239\)](#).

#### Water Removal Coef.

$\theta$  in [Equation 1.32 \(p. 239\)](#).

### 2.6.3.4. Specifying Catalyst Layer Properties for the Anode

**Figure 2.7: The Anode Tab of the PEM Fuel Cell Model Dialog Box with TPB Layer (Catalyst) Selected**



- Under **Anode Zone Type**, select **TPB Layer (Catalyst)**.
- From the **Zone(s)** selection list, select a corresponding zone. If you are modeling a fuel cell stack, then you must select *all* zones of a particular type as a group.
- From the **Solid Material** drop-down list, select the appropriate material. You can use the **Create/Edit Materials** dialog box to customize solid materials. Note that for the **Electrical Conductivity**, you can only select a constant value in the **Create/Edit Materials** dialog box. The solid electrical conductivity value is the diffusivity of the solid phase potential in the solid zones.
- Specify values for the following parameters:

#### Porosity

is  $\varepsilon$  in [Equation 1.38 \(p. 240\)](#).

#### Absolute Permeability

is  $K$  in [Equation 1.27 \(p. 238\)](#).

**Surface/Volume Ratio**

is the specific active surface area  $\zeta$  in [Equation 1.5 \(p. 234\)](#).

**H-phobic Contact Angle**

is the hydrophobic contact angle  $\theta_c$  in [Equation 1.33 \(p. 239\)](#).

**Protonic Conduction Coefficient**

is the model constant  $\beta_{an}$  in [Equation 1.42 \(p. 241\)](#) (available only when the **Effective P-Conductivity in MEA** option is selected under the **Model** tab).

**Protonic Conduction Exponent**

is the model constant  $\omega_i$  in [Equation 1.41 \(p. 241\)](#) (available only when the **Effective P-Conductivity in MEA** option is selected under the **Model** tab).

**Ionomer Tortuosity**

is the  $\tau_a$  in [Equation 1.42 \(p. 241\)](#) (available only when the **Effective P-Conductivity in MEA** option is selected under the **Model** tab).

**Ionomer Volume Fraction**

is the  $\varsigma_a$  in [Equation 1.42 \(p. 241\)](#) (available only when the **Effective P-Conductivity in MEA** option is selected under the **Model** tab).

**Activation Energy for P-Cond**

is the  $E_i$  in [Equation 1.41 \(p. 241\)](#) (available only when the **Effective P-Conductivity in MEA** option is selected under the **Model** tab).

**Activation Energy for J\_ref**

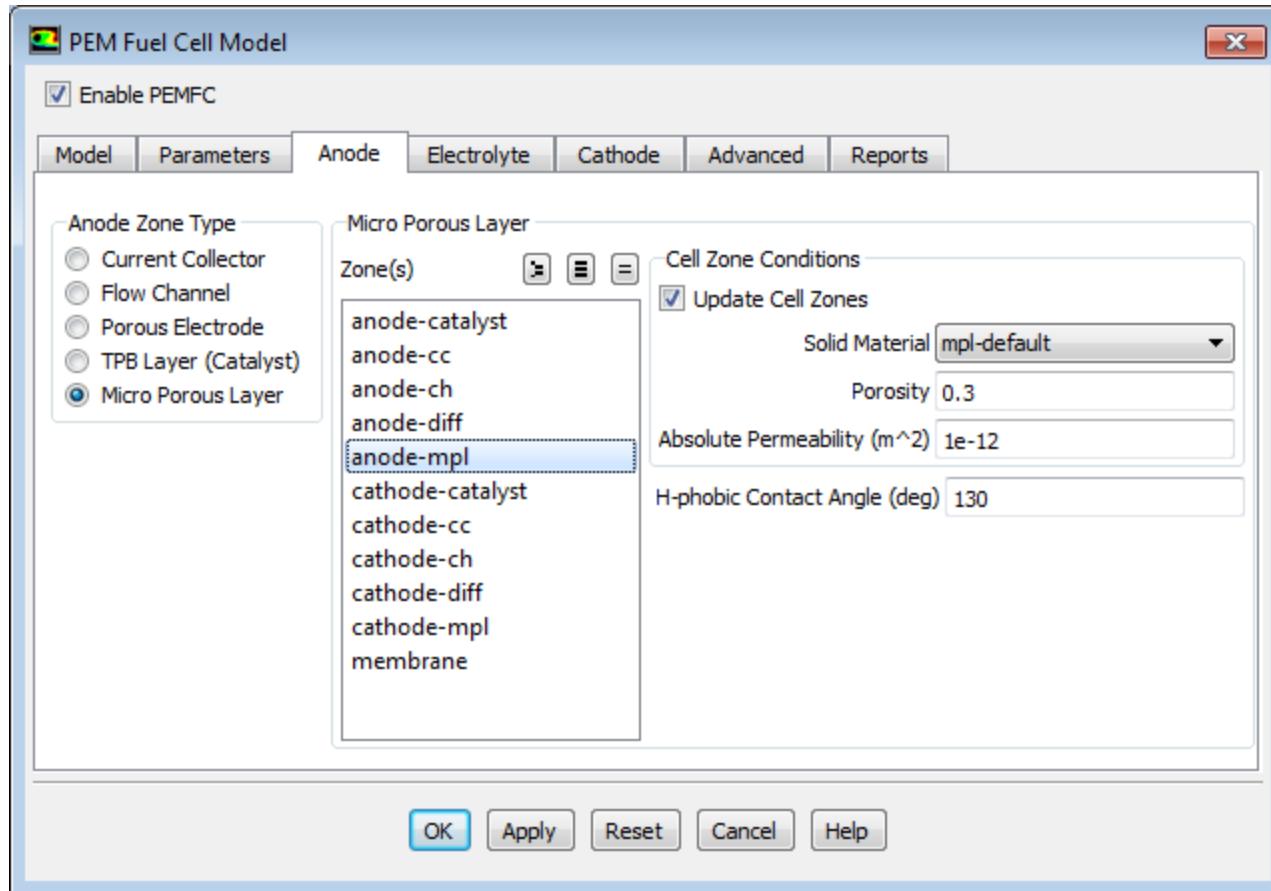
is the  $E_{an}$  in [Equation 1.9 \(p. 235\)](#).

**Reference Temperature Temperature for J\_ref**

is the  $T_{an}^{ref}$  in [Equation 1.9 \(p. 235\)](#).

### 2.6.3.5. Specifying Micro Porous Layer (Optional) Properties for the Anode

Figure 2.8: The Anode Tab of the PEM Fuel Cell Model Dialog Box with Micro Porous Layer Selected



- Under **Anode Zone Type**, select **Micro Porous Layer**.
- From the **Zone(s)** selection list, select a corresponding zone. If you are modeling a fuel cell stack, then you must select *all* zones of a particular type as a group.
- From the **Solid Material** drop-down list, select the appropriate material. You can use the **Create/Edit Materials** dialog box to customize solid materials. Note that for the **Electrical Conductivity**, you can only select a constant value in the **Create/Edit Materials** dialog box. The solid electrical conductivity value is the diffusivity of the solid phase potential in the solid zones.
- Specify values for the following parameters:

#### Porosity

is  $\varepsilon$  in [Equation 1.38 \(p. 240\)](#).

#### Absolute Permeability

is  $K$  in [Equation 1.27 \(p. 238\)](#).

#### H-phobic Contact Angle

hydrophobic contact angle  $\theta_c$  in [Equation 1.33 \(p. 239\)](#).

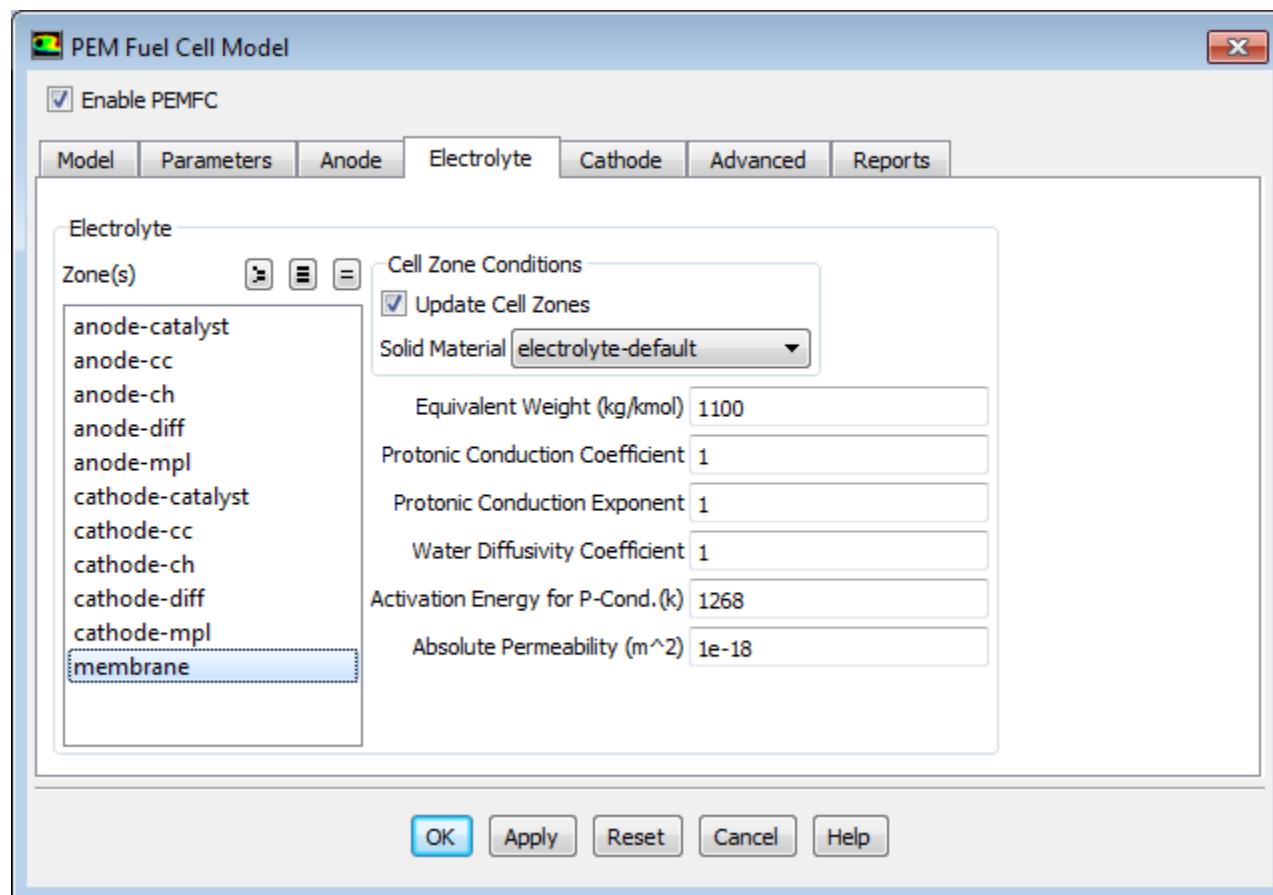
### 2.6.3.6. Specifying Cell Zone Conditions for the Anode

For each case of the anode's current collector, diffusion layer, micro porous layer, and catalyst layer, you assign a solid material and/or set the porosity and the viscous resistance. These settings represent setting a cell zone condition. With the **Update Cell Zones** option turned on (the default setting), this cell zone condition is applied to all selected zones in the **Zone(s)** list. If you want to set the cell zone conditions for each zone individually (using the **Cell Zone Conditions** task page), you should turn off the **Update Cell Zones** option.

### 2.6.4. Specifying Electrolyte/Membrane Properties

You can use the **Electrolyte** tab of the **PEM Fuel Cell Model** dialog box to specify zones and properties of the electrolyte/membrane portion of the fuel cell.

**Figure 2.9: The Electrolyte Tab of the PEM Fuel Cell Model Dialog Box**



- From the **Zone(s)** selection list, select a corresponding zone. If you are modeling a fuel cell stack, then you must select *all* zones of a particular type as a group.
- From the **Solid Material** drop-down list, select the appropriate material. You can use the **Create/Edit Materials** dialog box to customize solid materials.
- Specify values for the following parameters:

#### Equivalent Weight

is the *EW* in [Equation 1.23 \(p. 237\)](#) and [Equation 1.24 \(p. 237\)](#).

**Protonic Conduction Coefficient**

is the  $\beta_{mem}$  in [Equation 1.42 \(p. 241\)](#).

**Protonic Conduction Exponent**

is the  $\omega_i$  in [Equation 1.41 \(p. 241\)](#).

**Water Diffusivity Coefficient**

is the  $\eta_\lambda$  in [Equation 1.43 \(p. 241\)](#).

**Activation Energy for P-Cond**

is the  $E_i$  in [Equation 1.41 \(p. 241\)](#) (available only when the **Effective P-Conductivity in MEA** option is selected under the **Model** tab).

**Absolute Permeability**

is the  $K$  in [Equation 1.27 \(p. 238\)](#).

Note that the PEMFC model allows you to model the electrolyte/membrane as a solid zone only. However, it still allows for dissolved water, liquid or capillary pressure, and the ionic current to pass through.

### 2.6.4.1. Specifying Cell Zone Conditions for the Membrane

When you assign a solid material to the membrane, you are setting a cell zone condition. With the **Update Cell Zones** option turned on (the default setting), this cell zone condition is applied to all selected zones in the **Zone(s)** list. If you want to set the cell zone conditions for each zone individually (using the **Cell Zone Conditions** task page), you should turn off the **Update Cell Zones** option.

### 2.6.5. Specifying Cathode Properties

You can use the **Cathode** tab of the **PEM Fuel Cell Model** dialog box to specify zones and properties of the current collector, the flow channel, the diffusion layer, the micro porous layer, and the catalyst layer for the cathode portion of the fuel cell.

#### 2.6.5.1. Specifying Current Collector Properties for the Cathode

The procedure for specifying the current collector properties for the cathode is similar to that for the anode. For specific steps, refer to [Specifying Current Collector Properties for the Anode \(p. 255\)](#) for details and then substitute "cathode" for "anode" where appropriate.

#### 2.6.5.2. Specifying Flow Channel Properties for the Cathode

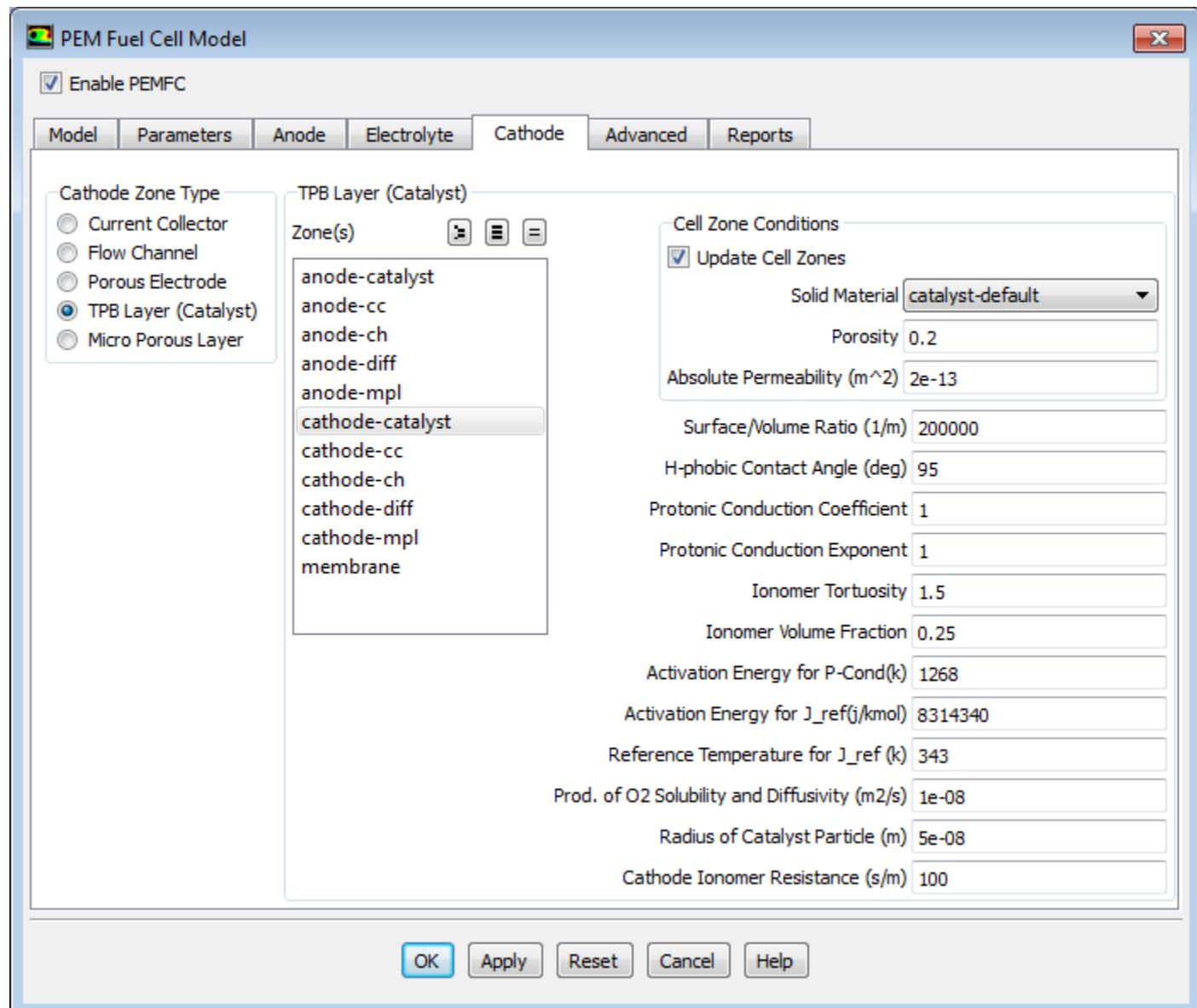
The procedure for specifying the flow channel properties for the cathode is similar to that for the anode. For specific steps, refer to [Specifying Flow Channel Properties for the Anode \(p. 256\)](#) and then substitute "cathode" for "anode" where appropriate.

#### 2.6.5.3. Specifying Porous Electrode Properties for the Cathode

The procedure for specifying the porous electrode properties for the cathode is similar to that for the anode. For specific steps, refer to [Specifying Porous Electrode Properties for the Anode \(p. 257\)](#) and then substitute "cathode" for "anode" where appropriate.

## 2.6.5.4. Specifying Catalyst Layer Properties for the Cathode

**Figure 2.10: The Cathode Tab of the PEM Fuel Cell Model Dialog Box with TPB Layer (Catalyst) Selected**



- Under **Cathode Zone Type**, select **TPB Layer (Catalyst)**.
- From the **Zone(s)** selection list, select a corresponding zone. If you are modeling a fuel cell stack, then you must select *all* zones of a particular type as a group.
- From the **Solid Material** drop-down list, select the appropriate material. You can use the **Create/Edit Materials** dialog box to customize solid materials. Note that for the **Electrical Conductivity**, you can only select a constant value in the **Create/Edit Materials** dialog box. The solid electrical conductivity value is the diffusivity of the solid phase potential in the solid zones.
- Specify values for the following parameters:

### Porosity

is  $\varepsilon$  in [Equation 1.38 \(p. 240\)](#).

**Absolute Permeability**

is  $K$  in [Equation 1.27 \(p. 238\)](#).

**Surface/Volume Ratio**

is the specific active surface area  $\zeta$  in [Equation 1.6 \(p. 234\)](#).

**H-phobic Contact Angle**

is the hydrophobic contact angle  $\theta_c$  in [Equation 1.33 \(p. 239\)](#).

**Protonic Conduction Coefficient**

is the model constant  $\beta_{ca}$  in [Equation 1.42 \(p. 241\)](#) (available only when the **Effective P-Conductivity in MEA** option is selected under the **Model** tab).

**Protonic Conduction Exponent**

is the model constant  $\omega_i$  in [Equation 1.41 \(p. 241\)](#) (available only when the **Effective P-Conductivity in MEA** option is selected under the **Model** tab).

**Ionomer Tortuosity**

is the  $\tau_c$  in [Equation 1.42 \(p. 241\)](#) (available only when the **Effective P-Conductivity in MEA** option is selected under the **Model** tab).

**Ionomer Volume Fraction**

is the  $\varsigma_c$  in [Equation 1.42 \(p. 241\)](#) (available only when the **Effective P-Conductivity in MEA** option is selected under the **Model** tab).

**Activation Energy for P-Cond**

is the  $E_i$  in [Equation 1.41 \(p. 241\)](#) (available only when the **Effective P-Conductivity in MEA** option is selected under the **Model** tab).

**Activation Energy for J\_ref**

is the  $E_{cat}$  in [Equation 1.10 \(p. 235\)](#).

**Reference Temperature Temperature for J\_ref**

is the  $T_{cat}^{ref}$  in [Equation 1.10 \(p. 235\)](#).

Besides the parameters for the catalyst layer listed above, you can also specify the following parameters for the [The Cathode Particle Model \(p. 235\)](#):

**Prod. Of O<sub>2</sub> Solubility and Diffusivity**

is  $K_w D_w$  in [Equation 1.16 \(p. 236\)](#).

**Radius of Catalyst Particle**

is  $r_p$  in [Equation 1.16 \(p. 236\)](#).

**Cathode Ionomer Resistance**

is  $\mathfrak{R}_{ion}$  in [Equation 1.15 \(p. 235\)](#).

## 2.6.5.5. Specifying Micro Porous Layer (Optional) Properties for the Cathode

The procedure for specifying the catalyst layer properties for the cathode is similar to that for the anode. For specific steps, refer to [Specifying Micro Porous Layer \(Optional\) Properties for the Anode \(p. 260\)](#) and then substitute “cathode” for “anode” where appropriate.

## 2.6.5.6. Specifying Cell Zone Conditions for the Cathode

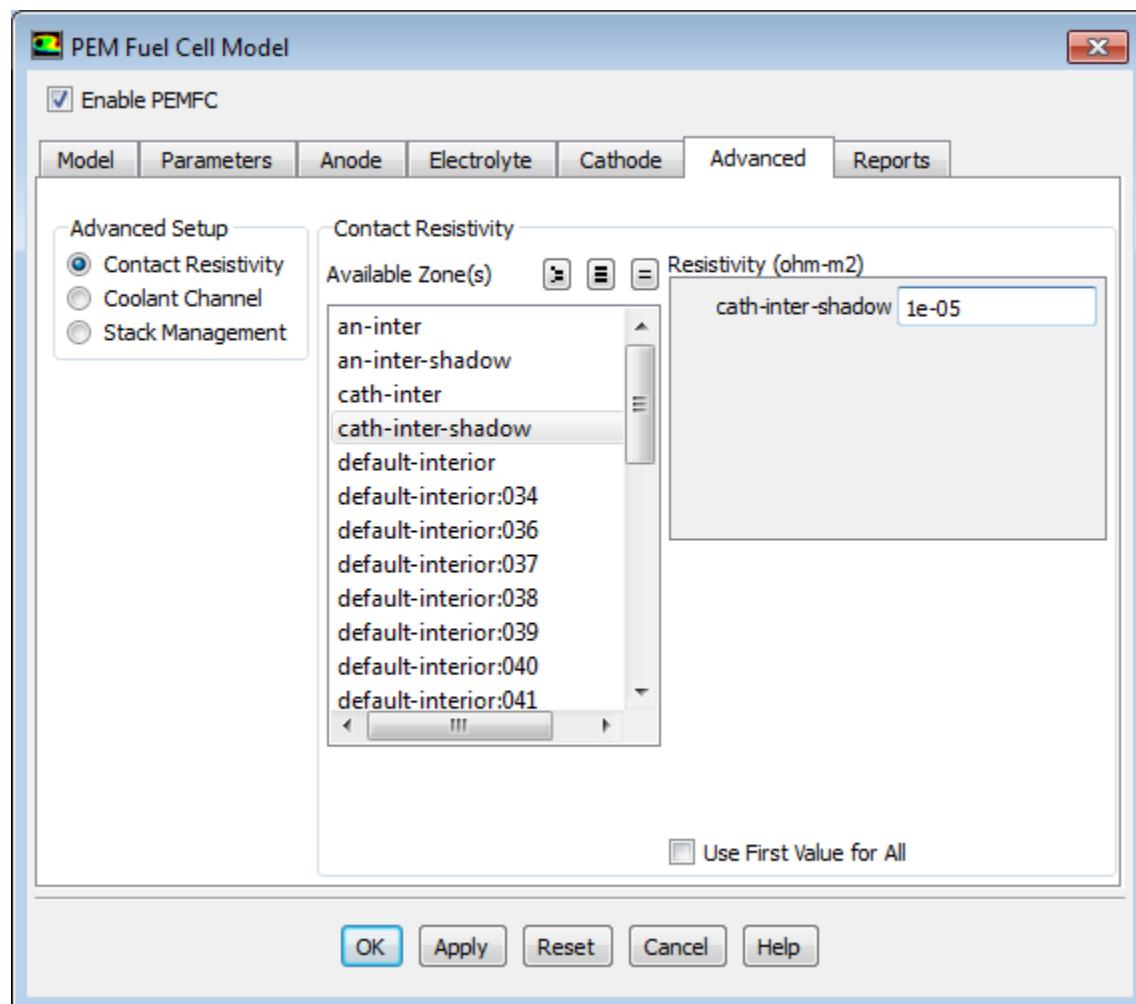
For each case of the cathode's current collector, diffusion layer, micro porous layer, and catalyst layer, you assign a solid material and/or set the porosity and the viscous resistance. These settings represent setting a cell zone condition. With the **Update Cell Zones** option turned on (the default setting), this cell zone condition is applied to all selected zones in the **Zone(s)** list. If you want to set the cell zone conditions for each zone individually (using the **Cell Zone Conditions** task page), you should turn off the **Update Cell Zones** option.

## 2.6.6. Setting Advanced Properties

You can use the **Advanced** tab of the **PEM Fuel Cell Model** dialog box to specify the contact resistivity for any material interface in the geometry, set parameters for coolant channels, and define fuel stack units for managing stacks of fuel cells.

### 2.6.6.1. Setting Contact Resistivities for the PEMFC Model

Figure 2.11: The Advanced Tab of the PEM Fuel Cell Model Dialog Box for Contact Resistivities



- Under **Advanced Setup**, select **Contact Resistivity**.

- From the **Available Zone(s)** selection list, select any number of corresponding interfaces. These zones are face zones over which a jump in electrical potential is caused by imperfect conduction.

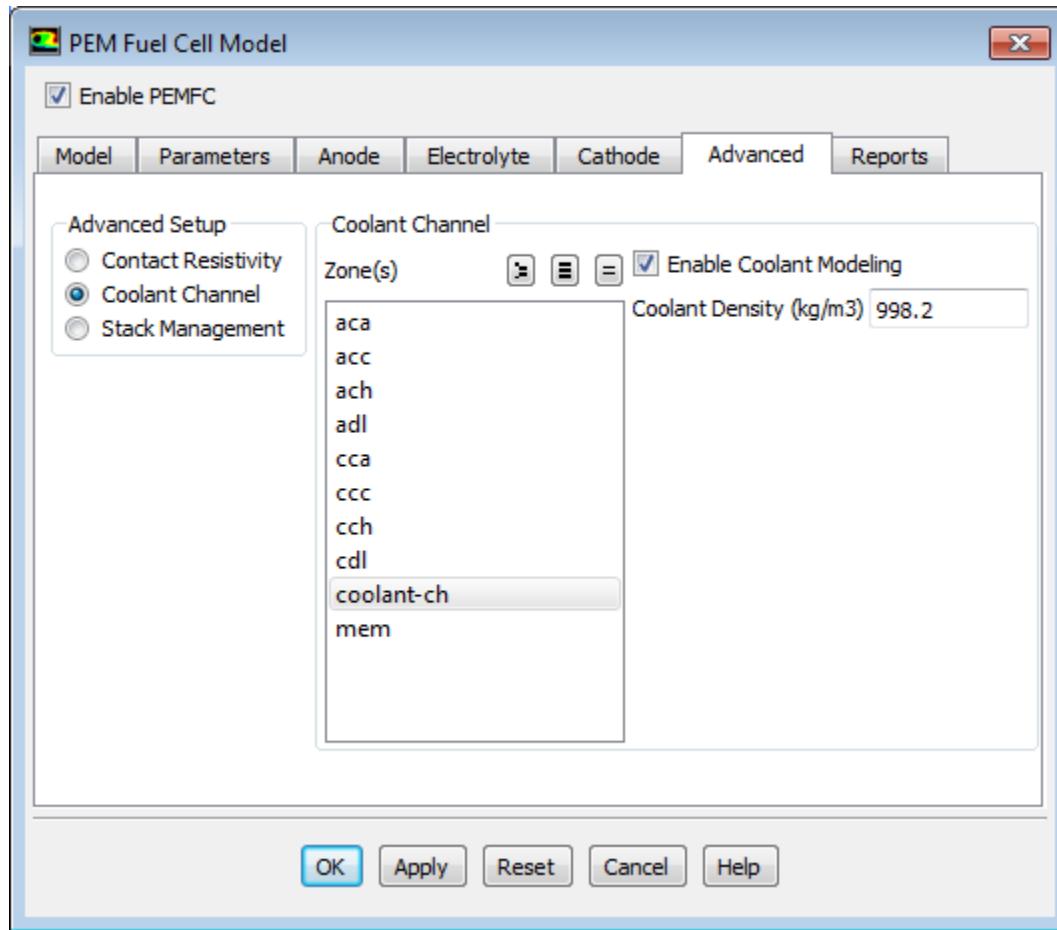
**Note**

The contact resistance will be applied only to wall and porous jump types of interfaces.

- Specify a value for the **Resistivity** for each specified zone.
- To simplify the input, you can choose to use the resistivity value of the first selected zone for all others as well by selecting the **Use First Value for All** option.

### 2.6.6.2. Setting Coolant Channel Properties for the PEMFC Model (Optional)

Figure 2.12: The Advanced Tab of the PEM Fuel Cell Model Dialog Box for the Coolant Channel

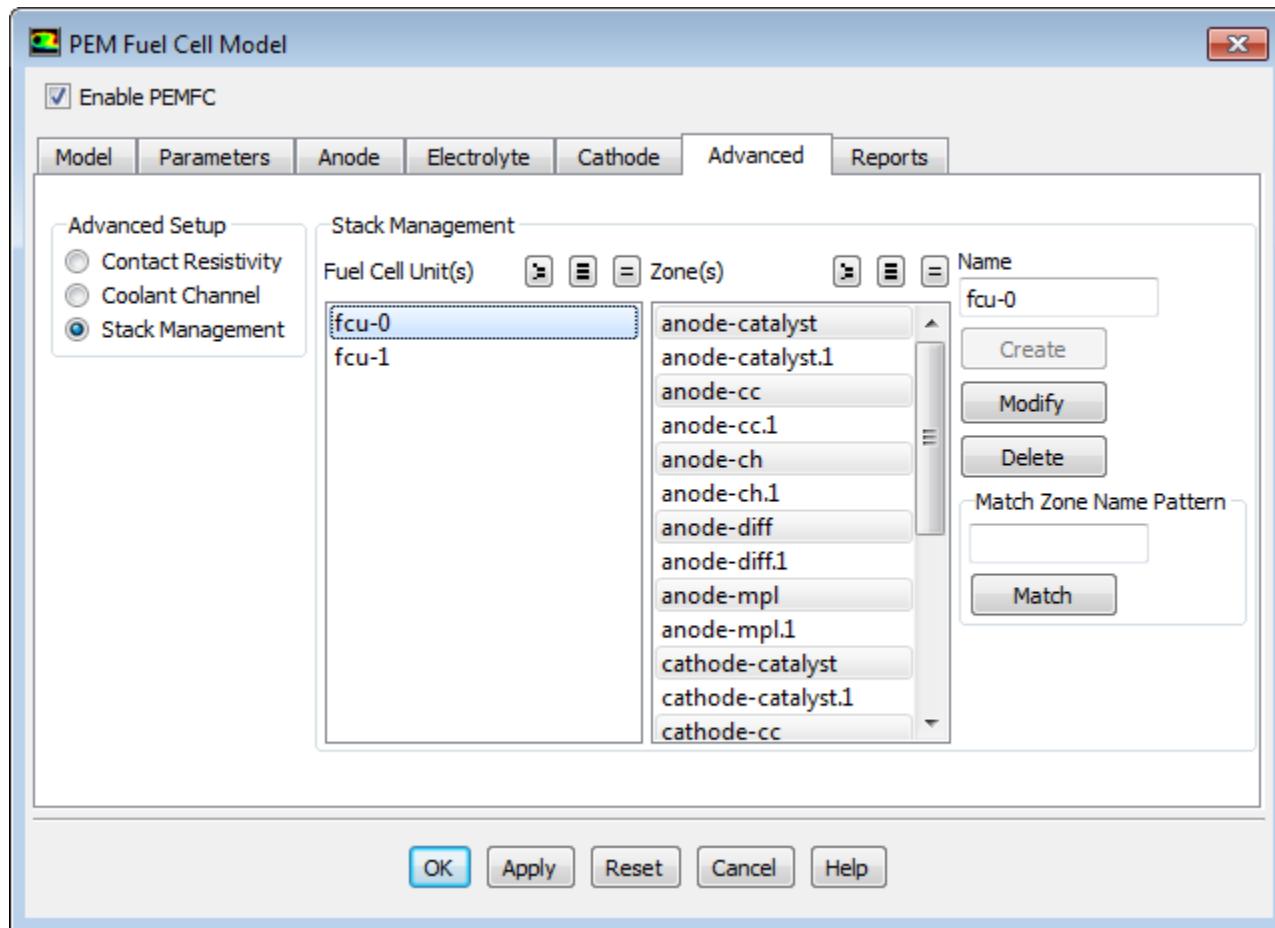


- Under **Advanced Setup**, select **Coolant Channel**.
- From the **Zone(s)** selection list, select any number of corresponding zones.
- Specify a value for the **Coolant Density**.
- To enable the coolant channel, select the **Enable Coolant Channel(s)** option. Amongst other settings, this will automatically create a new mixture material `pem+cool-mixture` consisting of hydrogen, oxygen, water-vapor, coolant-liquid, and nitrogen.

5. Using the **Create/Edit Materials** dialog box, set all other material properties for the new fluid coolant-liquid.
6. Specify the inlet and outlet conditions for cooling channels using standard boundary conditions dialog boxes.

### 2.6.6.3. Managing Stacks for the PEMFC Model

**Figure 2.13: The Advanced Tab of the PEM Fuel Cell Model Dialog Box for Stack Management**



The ANSYS Fluent PEMFC model allows you to model fuel cell stacks as well as individual fuel cells. In the **Advanced** tab of the **PEM Fuel Cell Model** dialog box, you can define *fuel cell units* for each fuel cell in a stack. A fuel cell unit consists of all zones of a single fuel cell in the stack.

#### Important

If you are only modeling a single fuel cell, then you do not need to set anything for **Stack Management** in the **Advanced** tab of the **PEM Fuel Cell Model** dialog box.

1. Select the **Advanced** tab of the **PEM Fuel Cell Model** dialog box.
2. Select **Stack Management** under **Advanced Setup**.
3. Since a fuel cell unit consists of all zones of a single fuel cell in the stack, select the corresponding zones from the **Zone(s)** list.

4. Create a new fuel cell unit by clicking the **Create** button. The new fuel cell is listed under **Fuel Cell Unit(s)** with a default name.
5. Edit a pre-existing fuel cell unit by selecting it in the **Fuel Cell Unit(s)** list. The zones in this fuel cell unit are automatically selected in the **Zone(s)** list. You can then modify the zones that comprise the fuel cell unit and/or change its name in the **Name** field and click **Modify** to save the new settings.
6. Remove a pre-existing fuel cell unit by selecting it in the **Fuel Cell Unit(s)** list and clicking the **Delete** button.

---

**Note**

When deleting a unit, make sure that the electrical connectivity remains intact.

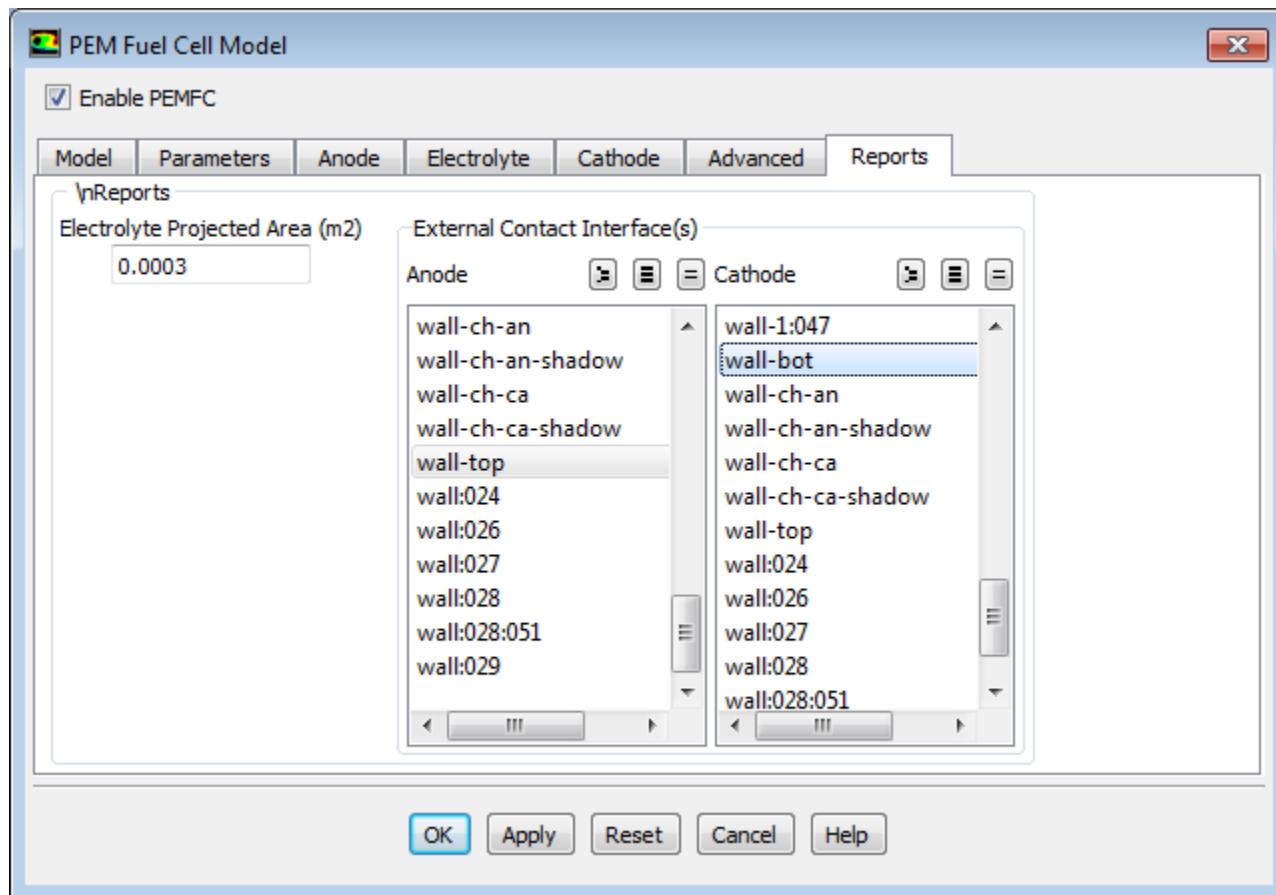
---

7. If your model contains many zone names, you can use the **Match Zone Name Pattern** field to specify a pattern to look for in the names of zones. Type the pattern in the text field and click **Match** to select (or deselect) the zones in the **Zones** list with names that match the specified pattern. You can match additional characters using \* and ?. For example, if you specify wall\*, all surfaces whose names begin with **wall** (for example, **wall-1**, **wall-top**) will be selected automatically. If they are all selected already, they will be deselected. If you specify wall?, all surfaces whose names consist of **wall** followed by a single character will be selected (or deselected, if they are all selected already).

For example, in a stack there are many fuel cells, say 10–100, each having at least 9 zones (current collector, gas channel, diffusion layer, and catalyst layer for both anode and cathode and a membrane). Additionally, there may be coolant channels, and it may be that for mesh construction reasons each of these physical zones is made up of more than one mesh zone. Even for small stacks, you can easily end up having hundreds of cell zones in an ANSYS Fluent mesh. Therefore, you may want to consider numbering the fuel cells in a stack and to use the assigned fuel cell number in the names of the mesh zones. When you set up your stacked fuel cell case, you would use the **Match Zone Name Pattern** field to pick all the zones belonging to a single fuel cell in the stack, rather than scrolling through the potentially very long list and selecting them manually.

## 2.6.7. Reporting on the Solution

You can use the **Reports** tab of the **PEM Fuel Cell Model** dialog box to set up parameters that will be useful in reporting data relevant to the fuel cell.

**Figure 2.14: The Reports Tab of the PEM Fuel Cell Model Dialog Box**

The **Electrolyte Projected Area** field requires the projected area of the Membrane Electrolyte Assembly (MEA) and is only used to calculate the average current density. The assembly consists of the membrane and the catalyst layers above and below the membrane.

The **External Contact Interface(s)** fields require the face zones that act as external electrical contact surfaces for the anode and the cathode.

These inputs are used to report cell voltage. For potentiostatic boundary conditions, this is the difference between the provided values, but for galvanostatic boundary conditions, the cell voltage is part of the solution.

## 2.7. PEMFC Model Boundary Conditions

The following boundary conditions need to be defined for the PEMFC simulation based on your problem specification:

- Anode Inlet
  - Mass flow rate
  - Temperature
  - Direction specification method
  - Mass fractions (for example, **h2**, and **h2o**)

- The coolant *must* be set to zero if coolant channels are enabled
- UDS-4 (Water Saturation in Channels) must be set to zero
- Cathode Inlet
  - Mass flow rate
  - Temperature
  - Direction specification method
  - Mass fractions (for example **o2**, **h2o**, and **n2**)
  - The coolant *must* be set to zero if coolant channels are enabled
  - UDS-4 (Water Saturation in Channels) must be set to zero
- Coolant Inlet (if any)
  - Mass flow rate
  - Temperature
  - Direction specification method
  - Coolant mass fraction set to 1
  - UDS-4 (Water Saturation in Channels) must be set to zero
- Pressure Outlets (all)  
Realistic backflow conditions.
- Terminal Anode
  - Temperature (or heat flux if known)
  - UDS-0 (electric potential) set to ground voltage
- Terminal Cathode
  - Temperature (or heat flux if known)
  - UDS-0 (electric potential) is set to the voltage of the cathode (if solving to constant voltage), or the UDS-0 (electric potential) flux is set to the current density in  $A/m^2$ (SI units) (if solving for constant current). Note that the sign of the UDS-0 flux on the cathode side is negative.

## 2.8. Solution Guidelines for the PEMFC Model

For potentiostatic boundary conditions, after initialization, solutions are calculated easily for cell voltages close to the open-circuit voltage. The same can be said for galvanostatic boundary conditions and low electric current. By lowering the cell voltage or by raising the average electric current, you can calculate subsequent stationary solutions.

In the event of convergence problems, it is recommended that you change the multigrid cycle to **F-cycle** with **BCGSTAB** (bi-conjugate gradient stabilized method) selected as the stabilization method for the species and the two potential equations. For the species and the user-defined scalar equations, it may be necessary to reduce the termination (criteria) of the multigrid-cycles to  $1\times10^{-3}$ . For stack simulations, the termination criterion may be reduced to  $\times10^{-7}$  for the two potential equations.

Also, it may be useful to turn off **Joule Heating** and **Reaction Heating** in the **PEM Fuel Cell Model** dialog box (in the **Model** tab) for the first few (approximately 5-10) iterations after initialization. This allows the two electric potentials to adjust from their initial values to more physical values, avoiding the possibility of extreme electrochemical reactions and electric currents that would in turn adversely impact the solution.

## 2.9. Postprocessing the PEMFC Model

You can perform postprocessing using standard ANSYS Fluent quantities and by using user-defined scalars and user-defined memory allocations. By default, the ANSYS Fluent PEMFC model defines several user-defined scalars and user-defined memory allocations, described in [Table 2.1: User-Defined Scalar Allocations \(p. 271\)](#) and [Table 2.2: User-Defined Memory Allocations \(p. 271\)](#).

**Table 2.1: User-Defined Scalar Allocations**

UDS 0	Electric Potential (solid phase potential) (Volts)
UDS 1	Protonic Potential (membrane phase potential) (Volts)
UDS 2	Capillary Pressure
UDS 3	Water Content
UDS 4	Liquid Saturation in Channels

**Table 2.2: User-Defined Memory Allocations**

UDM 0	X Current Flux Density ( $A/m^2$ )
UDM 1	Y Current Flux Density ( $A/m^2$ )
UDM 2	Z Current Flux Density ( $A/m^2$ )
UDM 3	Current Flux Density Magnitude ( $A/m^2$ )
UDM 4	Ohmic Heat Source ( $W/m^3$ )
UDM 5	Reaction Heat Source ( $W/m^3$ )
UDM 6	Overpotential (Volts)
UDM 7	Evaporation to Vapor Phase Change Source ( $kg/m^3\text{-s}$ )
UDM 8	Osmotic Drag Coefficient
UDM 9	Water Activity
UDM 10	Equilibrium Water Content
UDM 11	Absorption from Vapor phase change source ( $kg/m^3\text{-s}$ )
UDM 12	Absorption from Liquid Phase Change Source ( $kg/m^3\text{-s}$ )
UDM 13	Transfer Current ( $A/m^3$ )
UDM 14	Liquid Saturation in Porous Media
UDM 15	GDL Liquid Removal ( $kg/m^3\text{-s}$ )
UDM 16	Osmotic Drag Source ( $kg/m^3\text{-s}$ )

UDM 17	Capillary Pressure (Pa)
UDM 18	Pressure Gradient Source (kg/m <sup>3</sup> -s)

You can obtain this list by opening the **Execute On Demand** dialog box and pulling down the **Function** drop-down list.

 **User-Defined** → **User-Defined** → **Execute on Demand...**

and access the execute-on-demand function called `list_pemfc_udf`.

Alternatively, you can view the listing that appears when you first load your PEMFC case, or you can type `list_pemfc_udf` in the text user interface and the listing will appear in the console window.

---

### Note

- For field variables that are stored in UDM, use the corresponding variables for postprocessing. Postprocessing the UDM itself is not recommended.
  - For reviewing simulation results in CFD Post for cases involving UDM or UDS, export the solution data as CFD-Post-compatible file (.cdat) in Fluent and then load this file into CFD-Post.
- 

### Important

When you load older PEM Fuel Cell cases into ANSYS Fluent, and you are monitoring a UDS using volume or surface monitors, make sure you re-visit the corresponding monitors dialog box (for example, the **Volume Monitor** or the **Surface Monitor** dialog box) to verify that the correct UDS name is used for the appropriate monitor.

---

## 2.10. User-Accessible Functions

As noted in [Properties \(p. 240\)](#), you can directly incorporate your own formulations and data for the properties of the fuel cell membrane using the `pemfc_user.c` source code file.

The following listing represents a description of the contents of the `pemfc_user.c` source code file:

`real Get_P_sat(real T, cell_t c, Thread *t)`

Returns the value of the water vapor saturation pressure as a function of temperature ([Equation 1.45 \(p. 242\)](#)) or other user-specified values.

`real Water_Activity(real P, real T, cell_t c, Thread *t)`

Returns the value of water activity ([Equation 1.26 \(p. 237\)](#)).

`real Osmotic_Drag_Coefficient(cell_t c, Thread *t)`

Returns the value of the osmotic drag coefficient ([Equation 1.44 \(p. 242\)](#)).

`real Membrane_Conductivity(real lam, cell_t c, Thread *t)`

Returns the value of the protonic conductivity in MEA ([Equation 1.41 \(p. 241\)](#)).

---

```
real Water_Content_Diffusivity(real lam, real T, real mem_mol_dens-
ity,cell_tc,Thread*t)
```

Returns the value of the water content diffusivity in the MEA (Equation 1.43 (p. 241)).

```
real Gas_Diffusivity(cell_t c, Thread *t, int j_spe)
```

Returns the value of the gaseous species diffusivities in the channels, gas diffusion layers, micro porous layers, and catalysts (Equation 1.38 (p. 240)).

```
real MCD_Gas_Diffusivity(cell_t c, Thread *t, int i)
```

Returns the tortuosity-corrected value of the gas species diffusion coefficients computed with the multicomponent diffusion option (Equation 1.40 (p. 241)).

```
real Compute_Js(real s)
```

Compute the Leverett function; namely, the  $p_c$ - $s$  relation (Equation 1.33 (p. 239)).

```
Real Capillary_P_Diffusivity(real sat, real K_abs, cell_t c, Thread *t)
```

Returns the  $\rho_l KK_r / \mu_l$  in Equation 1.27 (p. 238).

```
real Anode_AV_Ratio(cell_t c, Thread *t)
```

Returns the value of the specific active surface area ( $\zeta_{an}$  in Equation 1.5 (p. 234)) for the anode catalyst.

```
real Cathode_AV_Ratio(cell_t c, Thread *t)
```

Returns the value of the specific active surface area ( $\zeta_{cat}$  in Equation 1.6 (p. 234)) for the cathode catalyst.

```
real Anode_J_TransCoef(cell_t c, Thread *t)
```

Returns the value of the anode reaction reference current density  $\frac{\zeta_{an} j_{an}^{ref}}{([H_2]_{ref})^{\gamma_{an}}}$  used in Equation 1.5 (p. 234).

```
real Cathode_J_TransCoef(cell_t c, Thread *t)
```

Returns the value of the cathode reaction reference current density  $\frac{\zeta_{cat} j_{cat}^{ref}}{([O_2]_{ref})^{\gamma_{cat}}}$  used in Equation 1.6 (p. 234).

```
real Open_Cell_Voltage(cell_t c, Thread *t)
```

Returns the value of the half-cell potential  $U^0$  used in Equation 1.11 (p. 235) and Equation 1.12 (p. 235).

```
real Leakage_Current(cell_t c, Thread *t)
```

Returns the value of the total leakage current ( $I_l$  in Equation 1.46 (p. 242) through Equation 1.48 (p. 243)).

```
real resistance_in_channel (real sat)
```

Returns momentum resistance as a function of liquid saturation in gas channels.

```
void Set_UDS_Names(char uds[n_uds_required][STRING_SIZE])
```

Used to rename user defined scalars (UDSs). Note that the units of the user defined scalars cannot be changed.

```
void Set_UDS_Names(char uds[n_uds_required][STRING_SIZE])
{
    strncpy(uds[0], "Electric Potential", STRING_SIZE-1);
    strncpy(uds[1], "Protonic Potential", STRING_SIZE-1);
    strncpy(uds[2], "Cap. Pressure", STRING_SIZE-1);
    strncpy(uds[3], "Water Content", STRING_SIZE-1);
    strncpy(uds[4], "Liq. Saturation in Channels", STRING_SIZE-1);
}
```

If you want to change the names of UDSs, change the second argument of the `strncpy` functions, recompile and link the module as with any modification to `pemfc_user.c`. Note that `STRING_SIZE` is fixed in `pemfc.h` and should not be changed.

---

### Important

When you load older PEM Fuel Cell cases into ANSYS Fluent, and you are monitoring a UDS using volume or surface monitors, make sure you re-visit the corresponding monitors dialog box (for example, the **Volume Monitor** or the **Surface Monitor** dialog box) to make sure that the correct UDS name is used for the appropriate monitor.

---

```
void Set_UDM_Names(char udm[n_udm_required][STRING_SIZE])
```

Used to rename user defined memory (UDMs). Note that the units of user defined memory cannot be changed.

```
void Set_UDM_Names(char udm[n_udm_required][STRING_SIZE])
{
    strncpy(udm[0], "X Current Flux Density", STRING_SIZE-1);
    strncpy(udm[1], "Y Current Flux Density", STRING_SIZE-1);
    strncpy(udm[2], "Z Current Flux Density", STRING_SIZE-1);
    strncpy(udm[3], "Current Flux Density Magnitude", STRING_SIZE-1);
    strncpy(udm[4], "Ohmic Heat Source", STRING_SIZE-1);
    strncpy(udm[5], "Reaction Heat Source", STRING_SIZE-1);
    strncpy(udm[6], "Overpotential", STRING_SIZE-1);
    strncpy(udm[7], "Evaporation to Vapor", STRING_SIZE-1);
    strncpy(udm[8], "Osmotic Drag Coefficient", STRING_SIZE-1);
    strncpy(udm[9], "Water Activity", STRING_SIZE-1);
    strncpy(udm[10], "Equilibrium Water Content", STRING_SIZE-1);
    strncpy(udm[11], "Absorption from Vapor", STRING_SIZE-1);
    strncpy(udm[12], "Absorption from Liquid", STRING_SIZE-1);
    strncpy(udm[13], "Transfer Current", STRING_SIZE-1);
    strncpy(udm[14], "Liquid Saturation", STRING_SIZE-1);
    strncpy(udm[15], "GDL Liquid Removal", STRING_SIZE-1);
    strncpy(udm[16], "Osmotic Drag Source (PEM)", STRING_SIZE-1);
    strncpy(udm[17], "Cap. Pressure", STRING_SIZE-1);
    strncpy(udm[18], "Pressure Gradient Source", STRING_SIZE-1);
}
```

If you want to change the names of UDMs, change the second argument of the `strncpy` functions, recompile and link the module as with any modification to `pemfc_user.c`. Note that `STRING_SIZE` is fixed in `pemfc.h` and should not be changed.

---

### Important

When you load older PEM Fuel Cell cases into ANSYS Fluent, and you are monitoring a UDM using volume or surface monitors, make sure you re-visit the corresponding monitors dialog box (for example, the **Volume Monitor** or the **Surface Monitor** dialog box) to make sure that the correct UDM name is used for the appropriate monitor.

---

```
real electric_contact_resistance(face_t f, Thread *t, int ns)
```

Returns the value for the electrical contact resistance.

```
real Transfer_Current(real i_ref, real gamma, int species_i, real alpha_a,
real alpha_c, real *dRade, real *dRcde, Thread *t, cell_t c, real cathode_volume)
```

Computes the transfer current ( $A/m^3$ ), corresponding to  $R_{an}$  in [Equation 1.5 \(p. 234\)](#) and  $R_{cat}$  in [Equation 1.6 \(p. 234\)](#).

Inputs for this function include:

i_ref	effective transfer current coefficient, computed by Cathode_J_TransCoef(c, t) or Anode_J_TransCoef(c, t)
gamma	cathode or anode concentration exponent
species_i	species index used in fuel cells (for example i_o2, i_h2, i_h2o)
alpha_a	product of anode exchange coefficient and $\frac{F}{RT}$
alpha_c	product of cathode exchange coefficient and $\frac{F}{RT}$
t	current thread
c	current cell
cathode_volume	total volume of cathode catalyst layer

Outputs for this function include:

source	anode or cathode volumetric transfer current ( $R_{an}$ in Equation 1.5 (p. 234) or $R_{cat}$ in Equation 1.6 (p. 234))
*dRade	partial derivative of $R_{an}$ with respect to activation loss
*dRcde	partial derivative of $R_{cat}$ with respect to activation loss

#### **Thermal\_ctk\_pemfc(face\_t f, Thread \*t)**

Used to change the constant value of **Thermal Contact Resistance** set in the **Porous Jump** dialog box to a locally variable value.

#### **Phase\_Change\_const(cell\_t c, Thread \*t)**

Used to change the constant value of  $\gamma_{gl}$  used to compute the gas-liquid phase change rate.

For more information, see the following sections:

##### [2.10.1. Compiling the Customized PEMFC Source Code](#)

This section includes instructions on how to compile a customized PEM Fuel Cell user-defined module. Note that you can also refer to the file **INSTRUCTIONS-CLIENT** that comes with your distribution (see **addons/pemfc**).

#### **Important**

It is assumed that you have a basic familiarity with compiling user-defined functions (UDFs). For an introduction on how to compile UDFs, refer to the [Fluent Customization Manual](#).

You will first want to use a local copy of the **pemfc** directory in the **addons** directory before you re-compile the PEM Fuel Cell module.

### 2.10.1.1. Compiling the Customized Source Code Under Linux

1. Make a local copy of the pemfc directory. Do not create a symbolic link.
- 

#### Important

The custom version of the library must be named according to the convention used by ANSYS Fluent: for example, pemfc.

---

2. Change directories to the pemfc/src directory.
3. Make changes to the pemfc\_user.c file.
4. Edit the makefile located in the src/ directory and make sure that the FLUENT\_INC variable correctly refers to the current ANSYS Fluent installation directory. Be careful not to leave any trailing spaces when you make your changes.
5. Define the FLUENT\_ADDONS environment variable to correspond to your customized version of the PEM Fuel Cell module.
6. Change directories to the pemfc/ directory.
7. Issue the following make command:

```
make FLUENT_INC=[ansys_inc/v170/fluent] FLUENT_ARCH=[arch]
-f Makefile-client
```

where your\_arch is lnx86 on LINUX, or ultra on the Sun operating system, and so on.

The following example demonstrates the steps required to set up and run a customized version of the PEM Fuel Cell module that is located in a folder call home/sample:

- Make a directory (for example, mkdir -p /home/sample).
- Copy the default addon library to this location.

```
cp -RH [ansys_inc/v170/fluent]/fluent17.0.0/addons/pemfc
/home/sample/pemfc
```

- Using a text editor, make the appropriate changes to the pemfc\_user.c file located in /home/sample/pemfc/src/pemfc\_user.c
- Edit the makefile located in the src/ directory and make sure that the FLUENT\_INC variable correctly refers to the current ANSYS Fluent installation directory. Be careful not to leave any trailing spaces when you make your changes.
- Build the library.

```
cd /home/sample/pemfc
make FLUENT_INC=[ansys_inc/v170/fluent] FLUENT_ARCH=[arch]
-f Makefile-client
```

- Set the FLUENT\_ADDONS environment variable (using CSH, other shells will differ).

```
setenv FLUENT_ADDONS /home/sample
```

- Start ANSYS Fluent and load the customized module using the text interface command.

### **2.10.1.2. Compiling the Customized Source Code under Windows**

1. Open **Visual Studio.NET** at the DOS prompt.
2. Make sure that the \$FLUENT\_INC environment variable is correctly set to the current ANSYS Fluent installation directory (for example, ANSYS Inc\v170\fluent).
3. Make a local copy of the pemfc folder. Do not create a shortcut.
4. Enter the pemfc\src folder.
5. Make changes to the pemfc\_user.c file.
6. Define the FLUENT\_ADDONS environment variable to correspond to your customized version of the PEM Fuel Cell module.
7. Return to the pemfc folder.
8. Issue the following command in the command window:

```
nmake /f makefile_master-client.nt
```

## **2.11. Using the PEMFC Text User Interface**

All of the features for the PEMFC model that are available through the graphical user interface are also available through text user interface (TUI) commands. The TUI allows text commands to be typed directly in the ANSYS Fluent console window where additional information can be extracted and processed for more advanced analysis.

Once the fuel cell module is loaded (see [Loading the PEMFC Module \(p. 246\)](#)), you can access the text user interface through the console window under `pemfc`. A listing of the various text commands is as follows:

```
enable-fc-model?
    Enable/disable PEMFC model

model-options
    Model options

model-parameters
    Model parameters

anode-setup/
    Anode setup

catalyst-layer
    Set catalyst layer

current-collector
    Set current collector

flow-channel
    Set flow channel
```

```
list-zones-briefly
  List zone names and IDs

micro-porous-layer
  Set micro-porous layer

porous-electrode
  Set porous electrode

cathode-setup/
  Cathode setup

catalyst-layer
  Set catalyst layer

current-collector
  Set current collector

flow-channel
  Set flow channel

list-zones-briefly
  List zone names and IDs

micro-porous-layer
  Set micro-porous layer

porous-electrode
  Set porous electrode

electrolyte-setup/
  Electrolyte setup

electrolyte-layer
  Set electrolyte layer

list-zones-briefly
  List zone names and IDs

advanced-setup/
  Advanced setup

list-zones-briefly
  List zone names and IDs

contact-resistivity
  Set contact resistivity

coolant-channel
  Set coolant channel

stack-management/
  Stack setup
```

---

```

list-fc-units
    List fuel cell units

list-zones-briefly
    List zone names and IDs

create-fc-unit
    Create fuel cell unit

modify-fc-unit
    Modify fuel cell unit

delete-fc-unit
    Delete fuel cell unit

set-stack-current-density
    Set the current density on the anode or cathode and modify the current solution to assist convergence. Note: Input here is in units of A/cm2. This is only available if the case contains valid data (for example, after initialization, iterating, or reading in data). For more information, see IV-Curve Calculations Using the Text Interface \(p. 279\).

set-stack-voltage
    Set the voltage difference in Volts between the anode and the cathode and modify the current solution to assist convergence. This is only available if the case contains valid data (for example, after initialization, iterating, or reading in data). For more information, see IV-Curve Calculations Using the Text Interface \(p. 279\).

reset-setup
    Reset the stack setup in case mistakes are made

submit-setup
    Submit the stack setup and makes the stack setup take effect

reports
    Set electrolyte project area and external contacts

```

## 2.11.1. IV-Curve Calculations Using the Text Interface

For valid case and data files, there are two text commands available to assist in the IV-curve calculation. These commands are **set-stack-voltage** (aliased as **ssv**) and **set-stack-current-density** (aliased as **ssc**), available from the PEMFC text command menu: /define/models/pemfc/advanced-setup/stack-management/.

For fuel cells, you either prescribe the voltage and obtain the total current delivered by the fuel cell as a result, or you specify the total current (via flux boundary conditions multiplied by the area) and obtain the voltage as part of the solution. The details of this IV-relation are specific for each single fuel cell and depend on mass and heat transport, electrochemistry and inlet conditions, outlet conditions, operating conditions, and any other parameter or material property involved in the calculation. The IV-curve is important for applications, because its product is the power delivered by the system.

As described earlier in this manual, you would start a new simulation from fairly static conditions, that is, high voltage/low current (which implies low species transport and low heat generation). After convergence, you typically may be interested in solutions for new electric boundary conditions, that is, either for a new cell/stack voltage or current.

In such cases, simply going to the **Boundary Conditions** task page and changing the value of the electric potential ( $uds=0$ ) boundary condition, typically allows only small changes, most notably for stacks. Otherwise the solution will not converge. This is where the set-stack-voltage and set-stack-current-density commands are important.

In addition to changing the boundary conditions (either to a prescribed voltage or current density), these commands process the current data in order to estimate the solution for the new boundary conditions. Because these commands modify the data, you are prompted to save your data, if you have not already done so.

Before going into details of the commands, here are some general remarks about electric potential boundary conditions.

For fixed voltage boundary conditions, both external contacts have a fixed value for the electric potential ( $uds=0$ ). The anode value will typically be zero, but it does not have to be. The cathode value will be larger than the anode value and the difference ( $V_{cathode} - V_{anode}$ ) is the positive cell/stack voltage.

For a fixed current boundary condition, one external contact has to have a fixed value and the other flux boundary conditions. As described earlier in the manual, typically, the anode will have a fixed (zero) value, and the cathode will be floating, however, you can also set the cathode to a fixed zero potential, yielding a floating negative anode potential.

The set-stack-voltage command sets the effective stack voltage, that is, the difference ( $V_{cathode} - V_{anode}$ ). For fixed voltage boundary conditions for the previous solution, boundary conditions on both boundaries are of type fixed value and then the cathode value will be changed accordingly. In the case of fixed current boundary conditions for the previous solution, the flux boundary condition will be changed to a fixed value boundary condition, and the value adjusted accordingly with respect to the other fixed value boundary condition.

The set-stack-current-density command sets the current density on one boundary to the desired value. Note that the input will be in  $\frac{A}{cm^2}$ , not  $\frac{A}{m^2}$  as you would normally have to enter in the **Boundary Conditions** task page. The reason for this is that average current densities reported in the text command interface are also in  $\frac{A}{cm^2}$ , and this makes it easier to choose the conditions you would like to prescribe next. Also, flux boundary conditions entered in the **Boundary Conditions** dialog box would have to have a positive sign on the anode side, and a negative sign on the cathode side. The input for the text interface command is just a positive number, signs are automatically accounted for.

For fixed current boundary conditions for the previous solution, the set-stack-current-density command changes the respective flux boundary condition accordingly. In the case of fixed voltage boundary conditions for the previous solution, the cathode side is chosen to be changed from a fixed value to a flux boundary condition with the new flux.

The two commands may be mixed in an IV-curve calculation. For the type of boundary condition setups currently described in this manual, boundary condition changes will consistently happen on the cathode side. However, if anode flux boundary conditions had been chosen initially, switching to fixed voltage boundary conditions by set-stack-voltage command and then back to fixed current boundary conditions by the set-stack-current-density command will then have flux boundary conditions on the cathode side. In this case, using the set-stack-current-density command exclusively will preserve the anode flux boundary condition setting.

---

---

## Chapter 3: Fuel Cell and Electrolysis Model Theory

---

This chapter presents the theoretical background for the Fuel Cell and Electrolysis modeling capabilities in ANSYS Fluent.

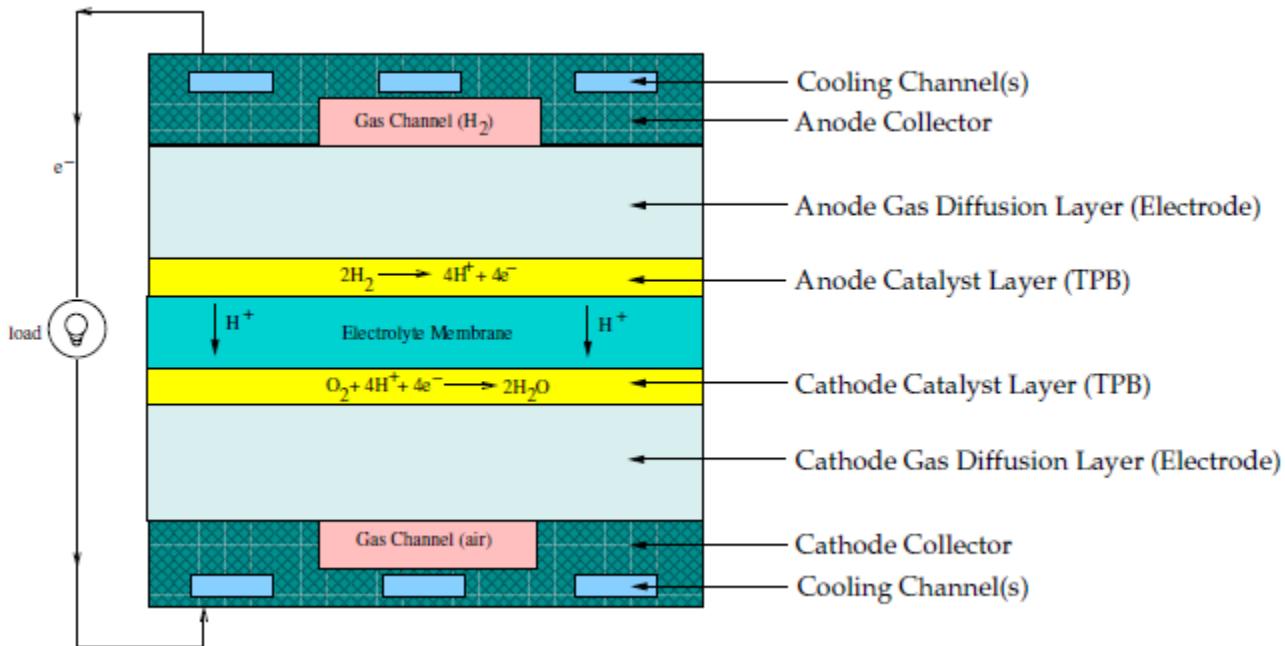
- 3.1. Introduction
- 3.2. Electrochemistry Modeling
- 3.3. Current and Mass Conservation
- 3.4. Heat Source
- 3.5. Liquid Water Formation, Transport, and its Effects (PEMFC Only)
- 3.6. Properties
- 3.7. Transient Simulations
- 3.8. Leakage Current (Cross-Over Current)

### 3.1. Introduction

The Fuel Cell and Electrolysis module (sometimes referred to as the Resolved Electrolyte module) is provided as an add-on module with the standard ANSYS Fluent licensed software.

A fuel cell is an energy conversion device that converts the chemical energy of fuel into electrical energy. With the Fuel Cell and Electrolysis Model, both the triple-phase boundary (TPB), also known as the catalyst layer, and the ionic conducting electrolyte (also known as the membrane in PEMFC terminology) are included in the computational domain. The Fuel Cell and Electrolysis module allows you to model PEMFC, SOFC, and high-temperature electrolysis.

To determine the physical domains that are included in the Fuel Cell and Electrolysis module, a schematic of a polymer electrolyte membrane fuel cell (PEMFC) is shown in [Figure 3.1: Schematic of a PEM Fuel Cell \(p. 282\)](#).

**Figure 3.1: Schematic of a PEM Fuel Cell**

Hydrogen flows into the fuel cell on the anode side. It diffuses through the porous gas diffusion layers and comes in contact with the catalyst layer. Here it forms hydrogen ions and electrons. The hydrogen ions diffuse through the polymer electrolyte membrane at the center, the electrons flow through the gas diffusion layer to the current collectors and into the electric load attached. Electrons enter the cathode side through the current collectors and the gas diffusion layer. At the catalyst layer on the cathode side, the electrons, the hydrogen ions, and the oxygen combine to form water.

In the Fuel Cell and Electrolysis Model in ANSYS Fluent, two electric potential fields are solved. One potential is solved in the electrolyte and the TPB catalyst layer. The other is solved in the TPB catalyst layer, the porous electrode, and the current collectors. The rate of electrochemical reactions are computed in the TPB layers at both the anode and the cathode. Based on the cell voltage that you prescribe, the current density value is computed. Alternatively, a cell voltage can be computed based on a prescribed average current density.

For more information, see the following sections:

- [3.1.1. Introduction to PEMFC](#)
- [3.1.2. Introduction to SOFC](#)
- [3.1.3. Introduction to Electrolysis](#)

### 3.1.1. Introduction to PEMFC

Over the last decade, the proton exchange membrane fuel cell (PEMFC) has emerged as a favored technology for auto transportation and power generation because it is compact, clean, runs at low temperature ( $<100^\circ\text{C}$ ), permits an adjustable power output, and can be started relatively rapidly. Hydrogen is supplied at the anode and air is supplied at the cathode. The following electrochemical reactions take place in the anode and cathode triple phase boundary (TPB) layers, respectively,

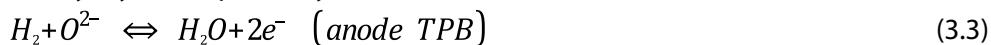


Electrons produced in the anode travel through an external circuit to the cathode, while protons ( $H^+$ ) travel through the membrane from the anode TPB to the cathode TPB, thereby forming an electrical circuit.

As more and more water is generated at the cathode, due to both osmotic drag and electrochemical reactions, water vapor pressure exceeds saturation pressure forming liquid water. The formation and transport of liquid water in the cathode is an important feature that can strongly influence cell performance of a PEMFC.

### 3.1.2. Introduction to SOFC

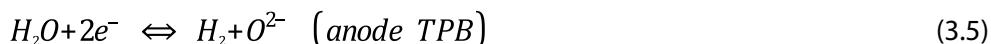
Unlike the low-temperature PEMFC, solid oxide fuel cells operate at very high temperatures ( $\sim 1000^\circ C$ ). Hydrogen is supplied at the anode, either directly or through internal reforming of another hydrocarbon fuel, and air is supplied at the cathode. The following electrochemical reactions take place in the anode and cathode triple phase boundary layers, respectively,



In the cathode TPB, oxygen is reduced to oxygen ions which are then conducted through the ceramic electrolyte to the anode TPB where they react with hydrogen to form water and release electrons. The electrons travel through an external circuit to a load and then back to the cathode to close the circuit.

### 3.1.3. Introduction to Electrolysis

There has been an increasingly strong need for the large-scale production of hydrogen as a secondary energy carrier. One of the cleaner and more efficient methods of producing hydrogen is to use high-temperature electrolysis to split water molecules. Therefore, electrolysis is essentially a reversed fuel cell process. Power is supplied to an electrolyzer to convert water vapor into hydrogen and oxygen. Water vapor is fed through the anode electrodes to the active electrolyte region. The following electrochemical reactions take place:



Note that, conventionally, the negative voltage is supplied to the cathode side in power consuming devices such as electrolyzers. ANSYS Fluent adopts the inverse notation where the negative voltage is supplied to the anode side whilst cathode remains positively charged. The main reason for this discrepancy is that the same infrastructure is used for both the electrolysis and fuel cells models. Usage of the terms "anode" and "cathode" in this manual and in the user interface should be interpreted according to conventions for power-supplying devices.

In electrolysis, the activation overpotentials have the opposite sign of what is used in fuel cells. This means that the cell voltage is higher than the open circuit voltage, since power is added to overcome the activation overpotentials. The ionic conductivity in the electrolyte is typically a function of temperature, such as in the case of SOFC. And it is pointed out here that, for an electrolyzer, high thermodynamic efficiency can be achieved only at a high operating temperature ( $>500^\circ C$ ). Because of this, the flow field is in vapor phase only and is handled as such within ANSYS Fluent.

### 3.2. Electrochemistry Modeling

At the center of the electrochemistry is the computation of the rates of the anodic and cathodic reactions. The electrochemistry model adopted in ANSYS Fluent is one that has been used by other groups ([4] (p. 355), [5] (p. 355), and [12] (p. 355)).

The driving force behind these reactions is the surface overpotential: the difference between the phase potential of the solid and the phase potential of the electrolyte/membrane. Therefore, two potential equations are solved for in the Fuel Cell and Electrolysis Model: one potential equation (Equation 3.7 (p. 284)) accounts for the electron transport  $e^-$  through the solid conductive materials and is solved in the TPB catalyst layer, the solid grids of the porous media, and the current collector; the other potential equation (Equation 3.8 (p. 284)) represents the protonic (that is, ionic) transport of  $H^+$  or  $O^{2-}$  and is solved in the TPB catalyst layer and the membrane. The two potential equations are as follows:

$$\nabla \cdot (\sigma_{sol} \nabla \phi_{sol}) + R_{sol} = 0 \quad (3.7)$$

$$\nabla \cdot (\sigma_{mem} \nabla \phi_{mem}) + R_{mem} = 0 \quad (3.8)$$

where

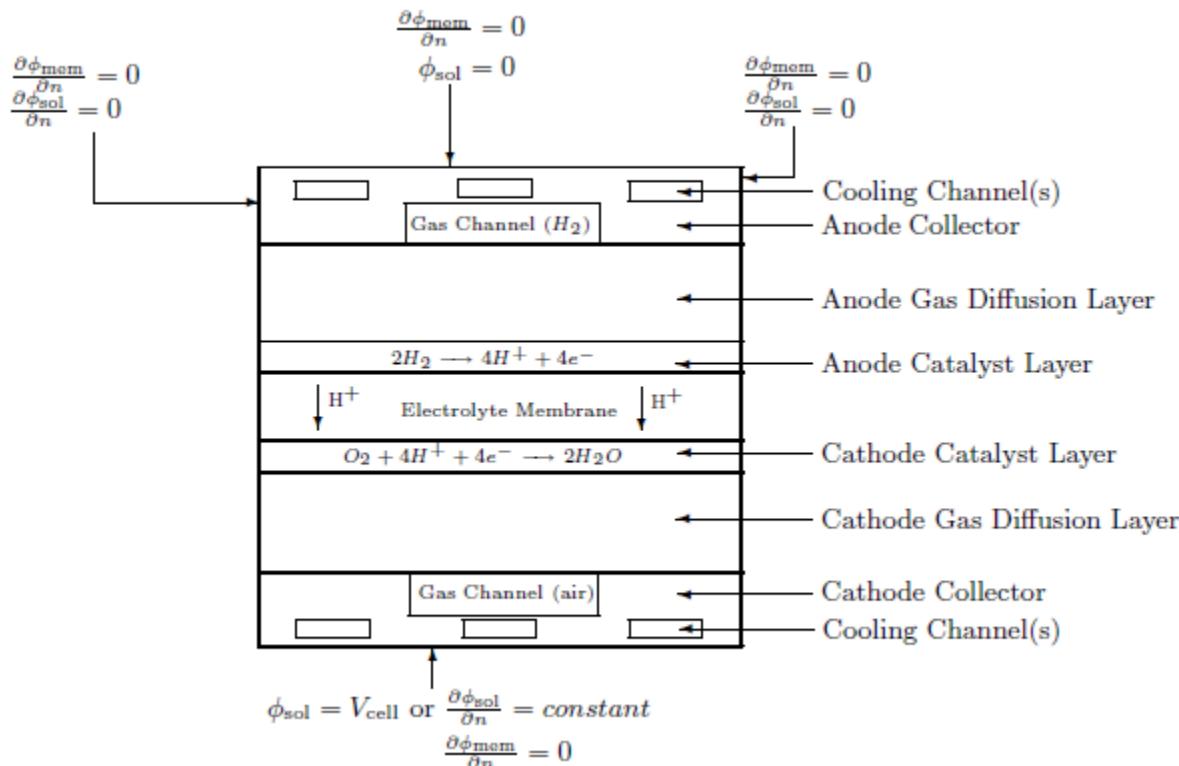
$\sigma$  = electrical conductivity (1/ohm-m)

$\phi$  = electric potential (volts)

$R$  = volumetric transfer current ( $A / m^3$ )

The following figure illustrates the boundary conditions that are used to solve for  $\phi_{sol}$  and  $\phi_{mem}$ .

**Figure 3.2: Boundary Conditions for the Electric Potential (Solid and Membrane) — PEM Fuel Cell**



There are two types of external boundaries: those that have an electrical current passing through them, and those that do not.

As no ionic current leaves the fuel cell through any external boundary, there is a zero flux boundary condition for the membrane phase potential,  $\phi_{mem}$ , on all outside boundaries.

For the solid phase potential,  $\phi_{sol}$ , there are external boundaries on the anode and the cathode side that are in contact with the external electric circuit and only through these boundaries passes the electrical current generated in the fuel cell. On all other external boundaries there is a zero flux boundary condition for  $\phi_{sol}$ .

On the external contact boundaries, we recommend to prescribe fixed values for  $\phi_{sol}$  (potentiostatic boundary conditions). If the anode side is set to zero, the (positive) value prescribed on the cathode side is the cell voltage. Specifying a constant flux (say on the cathode side) means to specify galvanostatic boundary conditions.

The transfer currents, or the source terms in [Equation 3.7 \(p. 284\)](#) and [Equation 3.8 \(p. 284\)](#), are nonzero only inside the catalyst layers and are computed as:

- For the potential equation in the solid phase,  $R_{sol} = -R_{an} (< 0)$  on the anode side and  $R_{sol} = +R_{cat} (> 0)$  on the cathode side.
- For the potential equation in the membrane phase,  $R_{mem} = +R_{an} (> 0)$  on the anode side and  $R_{mem} = -R_{cat} (< 0)$  on the cathode side.

The source terms in [Equation 3.7 \(p. 284\)](#) and [Equation 3.8 \(p. 284\)](#), also called the exchange current density ( $A/m^3$ ), have the following general definitions:

$$R_{an} = \left( \zeta_{an} j_{an}^{ref} \right) \left( \frac{[A]}{[A]_{ref}} \right)^{\gamma_{an}} \left( e^{\alpha_{an} F \eta_{an} / RT} - e^{-\alpha_{cat} F \eta_{an} / RT} \right) \quad (3.9)$$

$$R_{cat} = \left( \zeta_{cat} j_{cat}^{ref} \right) \left( \frac{[C]}{[C]_{ref}} \right)^{\gamma_{cat}} \left( -e^{+\alpha_{an} F \eta_{cat} / RT} + e^{-\alpha_{cat} F \eta_{cat} / RT} \right) \quad (3.10)$$

where

$j^{ref}$  = reference exchange current density per active surface area ( $A/m^2$ )

$\zeta$  = specific active surface area ( $1/m$ )

$[ ]$ ,  $[ ]_{ref}$  = local species concentration, reference value ( $kmol/m^3$ )

$\gamma$  = concentration dependence

$\alpha$  = transfer coefficient (dimensionless)

$F$  = Faraday constant ( $9.65 \times 10^7 C/kmol$ )

The above equation is the general formulation of the Butler-Volmer function. A simplification to this is the Tafel formulation that reads,

$$R_{an} = \left( \zeta_{an} j_{an}^{ref} \right) \left( \frac{[A]}{[A]_{ref}} \right)^{\gamma_{an}} \left( e^{\alpha_{an} F \eta_{an} / RT} \right) \quad (3.11)$$

$$R_{cat} = \left( \zeta_{cat} j_{cat}^{ref} \right) \left( \frac{[C]}{[C]_{ref}} \right)^{\gamma_{cat}} \left( e^{-\alpha_{cat} F \eta_{cat} / RT} \right) \quad (3.12)$$

By default, the Butler-Volmer function is used in the ANSYS Fluent Fuel Cell and Electrolysis Model to compute the transfer currents inside the catalyst layers.

In [Equation 3.9 \(p. 285\)](#) through [Equation 3.12 \(p. 285\)](#),  $[A]$  and  $[C]$  represent the molar concentration of the species upon which the anode and cathode reaction rates depend, respectively. For PEMFC and SOFC,  $A$  represents  $H_2$  and  $C$  represents  $O_2$ . For Electrolysis,  $A$  represents  $H_2O$  and  $C$  is 1.0 (which indicates that the cathode reaction does not depend on any species concentration).

The driving force for the kinetics is the local surface overpotential,  $\eta$ , also known as the *activation loss*. It is generally the difference between the solid and membrane potentials,  $\phi_{sol}$  and  $\phi_{mem}$ .

The gain in electrical potential from crossing from the anode to the cathode side can then be taken into account by subtracting the open-circuit voltage  $V_{oc}$  on the cathode side.

$$\eta_{an} = \phi_{sol} - \phi_{mem} \quad (3.13)$$

$$\eta_{cat} = \phi_{sol} - \phi_{mem} - V_{oc} \quad (3.14)$$

From [Equation 3.7 \(p. 284\)](#) through [Equation 3.14 \(p. 286\)](#), the two potential fields can be obtained.

### 3.3. Current and Mass Conservation

Species volumetric source terms ( $kg/m^3\text{-s}$ ) in the triple-phase boundaries due to electrochemical reactions for the PEMFC, SOFC, and Electrolysis, respectively, are:

#### For PEMFC:

$$S_{H_2} = -\frac{M_{w,H_2}}{2F} R_{an} < 0 \quad (3.15)$$

$$S_{O_2} = -\frac{M_{w,O_2}}{4F} R_{cat} < 0 \quad (3.16)$$

$$S_{H_2O} = \frac{M_{w,H_2O}}{2F} R_{cat} > 0 \quad (3.17)$$

#### For SOFC:

$$S_{H_2} = -\frac{M_{w,H_2}}{2F} R_{an} < 0 \quad (3.18)$$

$$S_{O_2} = -\frac{M_{w,O_2}}{4F} R_{cat} < 0 \quad (3.19)$$

$$S_{H_2O} = \frac{M_{w,H_2O}}{2F} R_{an} > 0 \quad (3.20)$$

#### For Electrolysis:

$$S_{H_2} = \frac{M_{w,H_2}}{2F} R_{an} > 0 \quad (3.21)$$

$$S_{O_2} = \frac{M_{w,O_2}}{4F} R_{cat} > 0 \quad (3.22)$$

$$S_{H_2O} = -\frac{M_{w,H_2O}}{2F} R_{an} < 0 \quad (3.23)$$

In the above equations,  $M_{w,H_2O}$ ,  $M_{w,O_2}$ , and  $M_{w,H_2}$  are the molecular mass of water, oxygen and hydrogen, respectively,  $F$  is the Faraday constant, and 2 and 4 are the numbers of electrons per mole of reactants and products.

Since the total electrical current produced in the cathode and the anode TPBs, respectively, is the same, we have the following equation for current conservation:

$$\int_{anode} R_{an} dV = \int_{cathode} R_{cat} dV \quad (3.24)$$

### 3.4. Heat Source

Additional volumetric sources to the thermal energy equation are present because not all chemical energy released in the electrochemical reactions can be converted to electrical work due to irreversibilities of the processes. The total source that goes to the thermal energy equation (that is, enthalpy) is:

$$S_h = h_{react} - R_{an,cat}\eta_{an,cat} + I^2R_{ohm} + h_L \quad (3.25)$$

where  $h_{react}$  is the net enthalpy change due to the electrochemical reactions,  $R_{an,cat}\eta_{an,cat}$  is the product of the transfer current and the overpotential in the anode or the cathode TPB,  $R_{ohm}$  is the ohmic resistivity of the conducting media, and  $h_L$  is the enthalpy change due to condensation/vaporization of water.

### 3.5. Liquid Water Formation, Transport, and its Effects (PEMFC Only)

Since PEM fuel cells operate under relatively low temperature ( $<100^\circ\text{C}$ ), the water vapor may condense to liquid water, especially at high current densities. While the existence of the liquid water keeps the membrane hydrated, it also blocks the gas diffusion passage, reduces the diffusion rate and the effective reacting surface area and hence the cell performance. To model the formation and transport of liquid water, ANSYS Fluent uses a saturation model based on [8] (p. 355), [6] (p. 355). In this approach, the liquid water formation and transport is governed by the following conservation equation for the volume fraction of liquid water,  $s$ , or the water saturation,

$$\frac{\partial(\varepsilon\rho_l s)}{\partial t} + \nabla \cdot (\rho_l \vec{V}_l s) = r_w \quad (3.26)$$

where  $\varepsilon$  is the porosity, the subscript  $l$  stands for liquid water, and  $r_w$  is the condensation rate that is modeled as,

$$r_w = \begin{cases} (1-s)c_r \frac{P_{wv}-P_{sat}}{RT} M_{h2o} & \text{if } (P_{wv} > P_{sat}) \\ s c_r \frac{P_{wv}-P_{sat}}{RT} M_{h2o} & \text{if } (P_{wv} < P_{sat}) \end{cases} \quad (3.27)$$

where  $-r_w$  is added to the water vapor equation, as well as the pressure correction (mass source). This term is applied only inside the catalyst and gas diffusion layers. The condensation rate constant is hardwired to  $c_r = 100\text{s}^{-1}$ . It is assumed that the liquid velocity,  $V_l$ , is equivalent to the gas velocity inside the gas channel (that is, a fine mist). Inside the highly-resistant porous zones, the use of the capillary diffusion term allows us to replace the convective term in Equation 3.26 (p. 287):

$$\frac{\partial(\varepsilon\rho_l s)}{\partial t} + \nabla \cdot \left[ \rho_l \frac{Ks^\gamma}{\mu_l} \frac{dp_c}{ds} \nabla s \right] = r_w \quad (3.28)$$

Depending on the wetting phase, the capillary pressure is computed as a function of  $s$  (the Leverett function),

$$p_c = \begin{cases} \frac{\sigma \cos \theta_c}{\left(\frac{K}{\varepsilon}\right)^{0.5}} (1.417(1-s) - 2.12(1-s)^2 + 1.263(1-s)^3) & \theta_c < 90^\circ \\ \frac{\sigma \cos \theta_c}{\left(\frac{K}{\varepsilon}\right)^{0.5}} (1.417s - 2.12s^2 + 1.263s^3) & \theta_c > 90^\circ \end{cases} \quad (3.29)$$

where  $\sigma$  is the surface tension (N/m),  $\theta_c$  is the contact angle and  $K$  the absolute permeability.

[Equation 3.26 \(p. 287\)](#) models various physical processes such as condensation, vaporization, capillary diffusion, and surface tension.

The clogging of the porous media and the flooding of the reaction surface are modeled by multiplying the porosity and the active surface area by  $(1-s)$ , respectively.

## 3.6. Properties

- Gas Phase Species Diffusivity

Gas phase species diffusivities can be computed either by using the dilute approximation method or by using the full multicomponent method. With the dilute approximation method, we have

$$D_i = \varepsilon^{1.5} (1-s)^{r_s} D_i^0 \left( \frac{p_0}{p} \right)^{\gamma_p} \left( \frac{T}{T_0} \right)^{\gamma_t} \quad (3.30)$$

where  $D_i^0$  is the mass diffusivity of species  $i$  at reference temperature and pressure ( $P_0, T_0$ ) [\[12\] \(p. 355\)](#). These reference values and the exponents ( $\gamma_p, \gamma_t$ ) as well as the exponent of pore blockage ( $r_s$ ) are defined in the Fuel Cell and Electrolysis user defined functions (UDF) as,

$$\begin{aligned} p_0 &= 101325 N/m^2 \\ T_0 &= 300 K \\ \gamma_p &= 1.0 \\ \gamma_t &= 1.5 \\ r_s &= 2.5 \end{aligned} \quad (3.31)$$

In addition to [Equation 3.30 \(p. 288\)](#), the ANSYS Fluent Fuel Cell and Electrolysis Model also contains a method to compute the gas phase species diffusion (a full multicomponent diffusion method with corrections to account for the porous media tortuosity):

$$D_{eff}^{ij} = \varepsilon^{1.5} D^{ii} \quad (3.32)$$

where  $D_{eff}^{ij}$  is the effective gas species diffusivity,  $\varepsilon$  is the porosity of the porous medium, and  $D^{ii}$  is the gas species mass diffusivity computed by the full multicomponent diffusion method (as described in [Full Multicomponent Diffusion](#) in the separate ANSYS Fluent [User's Guide](#)). Note that  $\varepsilon^{1.5}$  in [Equation 3.32 \(p. 288\)](#) is used to model the effect of tortuosity. While this is implemented as the default method in the Fuel Cell and Electrolysis Model, you can overwrite it with your own correction methods by using the user-modifiable routines that are provided.

Properties such as electrolyte phase electrical conductivity, water diffusivity, and the osmotic drag coefficient are evaluated as functions of the water content, using various correlations as suggested by [\[11\] \(p. 355\)](#). To capture the relevant physics of the problem, various properties of the membrane are incorporated into the model as default options. You can, however, directly incorporate your own formulations and data for these properties by editing the functions defined in the provided source code file called `pem_user.c` and compiling the code yourself. For more information, see [User-Accessible Functions \(p. 317\)](#).

- Electrolyte Phase (Ionic) Conductivity

For SOFC and high-temperature Electrolysis, the ionic conductivity in the electrolyte is modeled as a function of temperature, and, by default, is defined as:

$$\sigma_{ionic} = \frac{100}{0.3685 + 0.002838e^{(10300/T)}} \quad (3.33)$$

This is valid for temperatures ranging from 1073 K to 1373 K. You can implement your own models in the user-customizable UDF function `Electrolyte_Conductivity` in `pem_user.c`.

For PEMFC, the electrolyte (also called the membrane) phase conductivity is modeled as:

$$\sigma_{mem} = \beta (0.514\lambda - 0.326)^\omega e^{1268(\frac{1}{303} - \frac{1}{T})} \quad (3.34)$$

where  $\lambda$  is the water content. Two model constants,  $\beta$  and  $\omega$  are introduced in ANSYS Fluent for generality. [Equation 3.34 \(p. 289\)](#) becomes the original correlation from [\[11\] \(p. 355\)](#) when  $\beta=\omega=1$ .

- Osmotic Drag Coefficient (PEMFC)

$$n_d = 2.5 \frac{\lambda}{22} \quad (3.35)$$

- Back Diffusion Flux (PEMFC)

$$J_w^{diff} = -\frac{\rho_m}{M_m} M_{h_20} D_l \nabla \lambda \quad (3.36)$$

where  $\rho_m$  and  $M_m$  are the density and the equivalent weight of the dry membrane, respectively.

- Membrane Water Diffusivity (PEMFC)

$$D_l = f(\lambda) e^{2416(\frac{1}{303} - \frac{1}{T})} \quad (3.37)$$

- Water Content (PEMFC)

The water content,  $\lambda$ , that appears in the preceding property computations are obtained using Springer et al's correlation [\[11\] \(p. 355\)](#),

$$\begin{aligned} \lambda &= 0.043 + 17.18a - 39.85a^2 + 36a^3 \quad (a < 1) \\ \lambda &= 14 + 1.4(a-1) \quad (a > 1) \end{aligned} \quad (3.38)$$

here  $a$  is the water activity that is defined as,

$$a = \frac{P_{wv}}{P_{sat}} + 2s \quad (3.39)$$

where  $P_{wv}$  and  $P_{sat}$  are the water vapor pressure and the saturation pressure, respectively.

- Water Vapor Pressure

The water vapor pressure is computed based upon the vapor molar fraction and the local pressure,

$$P_{wv} = x_{H_2O} P \quad (3.40)$$

- Saturation Pressure

The saturation pressure is calculated, in terms of *atm*, as,

$$\begin{aligned} \log_{10} P_{sat} &= -2.1794 + 0.02953(T - 273.17) \\ &\quad - 9.1837 \times 10^{-5}(T - 273.17)^2 \\ &\quad + 1.4454 \times 10^{-7}(T - 273.17)^3 \end{aligned} \quad (3.41)$$

It is noted here that in [11] (p. 355), water activity is defined on the basis of total water or super-saturated water vapor. With phase change being invoked in the present two-phase model, 2s is added to the original formulation as suggested by [3] (p. 355).

## 3.7. Transient Simulations

Dynamics response to changes in operating conditions as a function of time can be modeled using the Fuel Cell and Electrolysis module. For example, a change in the cell voltage or current density, or inlet mass flow rates at the anode and/or the cathode. The procedure for setting up and solving transient Fuel Cell and Electrolysis problems are the same as that used for a normal ANSYS Fluent transient problem as discussed in the ANSYS Fluent User's Guide.

Assuming that the time scales associated with the electric fields are much smaller than those associated with the flow and thermal fields, the steady-state equations are retained for the two electric potentials, (that is, [Equation 3.7 \(p. 284\)](#) and [Equation 3.8 \(p. 284\)](#)). Transient terms in all other equations such as momentum transport, energy transport, species transport, liquid water transport, and membrane water content equations are activated.

## 3.8. Leakage Current (Cross-Over Current)

The leakage current,  $I_{leak}$  (A), models the overall effect of species cross-over from one electrode to another across the electrolyte. In addition to the source terms expressed by [Equation 3.15 \(p. 286\)](#) through [Equation 3.23 \(p. 286\)](#), the following extra volumetric source terms are added to the corresponding equations:

**For PEMFC and SOFC:**

$$S_{H_2} = -\frac{M_{w,H_2}}{2F} \cdot \frac{I_{leak}}{Vol_{anode}} \quad (3.42)$$

$$S_{O_2} = -\frac{M_{w,O_2}}{4F} \cdot \frac{I_{leak}}{Vol_{cathode}} \quad (3.43)$$

$$S_{H_2O} = -\frac{M_{w,H_2O}}{2F} \cdot \frac{I_{leak}}{Vol_{cathode}} \quad (3.44)$$

**For Electrolysis:**

$$S_{H_2} = \frac{M_{w,H_2}}{2F} \cdot \frac{I_{leak}}{Vol_{anode}} \quad (3.45)$$

$$S_{O_2} = \frac{M_{w,O_2}}{4F} \cdot \frac{I_{leak}}{Vol_{cathode}} \quad (3.46)$$

$$S_{H_2O} = -\frac{M_{w,H_2O}}{2F} \cdot \frac{I_{leak}}{Vol_{anode}} \quad (3.47)$$

Here,  $Vol_{cathode}$  and  $Vol_{anode}$  are the he volumes of the catalytic layers on the anode and cathode sides, and 2 and 4 are the numbers of electrons per mole of reactants and products.

---

## Chapter 4: Using the Fuel Cell and Electrolysis Model

---

The procedure for setting up and solving fuel cell problems using the Fuel Cell and Electrolysis Model is described in detail in this chapter. Refer to the following sections for more information:

- 4.1. Overview and Limitations
- 4.2. Geometry Definition for the Fuel Cell and Electrolysis Model
- 4.3. Installing the Fuel Cell and Electrolysis Model
- 4.4. Loading the Fuel Cell and Electrolysis Module
- 4.5. Setting Up the Fuel Cell and Electrolysis Module
- 4.6. Modeling Fuel Cells and Electrolysis
- 4.7. Modeling Current Collectors
- 4.8. Fuel Cell and Electrolysis Model Boundary Conditions
- 4.9. Solution Guidelines for the Fuel Cell and Electrolysis Model
- 4.10. Postprocessing the Fuel Cell and Electrolysis Model
- 4.11. User-Accessible Functions
- 4.12. Using the Fuel Cell and Electrolysis Text User Interface

### 4.1. Overview and Limitations

The ANSYS Fluent Fuel Cell and Electrolysis Model are made up of several user-defined functions (UDFs) and a graphical user interface. The potential fields are solved as user-defined scalars. The liquid water saturation,  $s$ , and the water content,  $\lambda$ , are also solved as user-defined scalars. The electrochemical reactions occurring on the catalyst are modeled through various source terms while other model parameters are handled through the user interface. The Fuel Cell and Electrolysis Model can be used in parallel ANSYS Fluent as well.

Note the following limitation when using the Fuel Cell and Electrolysis model:

- The anisotropic species diffusivity option is not compatible with the Fuel Cell and Electrolysis model.

### 4.2. Geometry Definition for the Fuel Cell and Electrolysis Model

Due to the fact that there are a number of different physical zones associated with the fuel cell, the following regions must be present in the fuel cell mesh:

- Anode flow channel
- Anode gas diffusion layer
- Anode catalyst layer
- Membrane layer
- Cathode catalyst layer
- Cathode gas diffusion layer
- Cathode flow channel

The following zones have to be identified, if present in the fuel cell mesh:

- Anode current collector
- Cathode current collector
- Coolant channel

## 4.3. Installing the Fuel Cell and Electrolysis Model

The Fuel Cell and Electrolysis Model is provided as an add-on module with the standard ANSYS Fluent licensed software. The module is installed with the standard installation of ANSYS Fluent in a directory called `addons/fuelcells` in your installation area. The Fuel Cell and Electrolysis Model consists of a UDF library and a pre-compiled scheme library, which must be loaded and activated before calculations can be performed.

## 4.4. Loading the Fuel Cell and Electrolysis Module

The Fuel Cell and Electrolysis module is loaded into ANSYS Fluent through the text user interface (TUI). The module can only be loaded after a valid ANSYS Fluent case file has been set or read. The text command to load the add-on module is

`define → models → addon-module`

A list of ANSYS Fluent add-on modules is displayed:

```
> /define/models/addon-module
Fluent  Addon Modules:
 0. None
 1. MHD Model
 2. Fiber Model
 3. Fuel Cell and Electrolysis Model
 4. SOFC Model with Unresolved Electrolyte
 5. Population Balance Model
 6. Adjoint Solver
 7. Battery Module
 8. MSMD Battery Model
 9. PEM Fuel Cell Model
Enter Module Number: [0] 3
```

Select the Fuel Cell and Electrolysis Model by entering the module number 3. During the loading process, a scheme library (containing the graphical and text user interface) and a UDF library (containing a set of user defined functions) are loaded into ANSYS Fluent.

## 4.5. Setting Up the Fuel Cell and Electrolysis Module

The Fuel Cell and Electrolysis Model can be used to model polymer electrolyte membrane fuel cells (PEMFC), solid oxide fuel cells (SOFC), and the process of high-temperature electrolysis. The following describes an overview of the procedure required in order to use the Fuel Cell and Electrolysis Model in ANSYS Fluent.

1. Start ANSYS Fluent.
2. Read the case file.
3. Scale the grid, if necessary.
4. Use the **Fuel Cell and Electrolysis Models** dialog box to define the fuel cell model parameters.

5. Define material properties.
6. Set the operating conditions.
7. Set the boundary conditions.
8. Start the calculations.
9. Save the case and data files.
10. Process your results.

### **Important**

The **Fuel Cell and Electrolysis Models** dialog box greatly simplifies the input of parameters and boundary conditions, but it does not replace the boundary conditions interface. Therefore, it is a good policy to start the setup with the **Fuel Cell and Electrolysis Models** dialog box and do the finishing steps for boundary conditions afterwards.

### **Important**

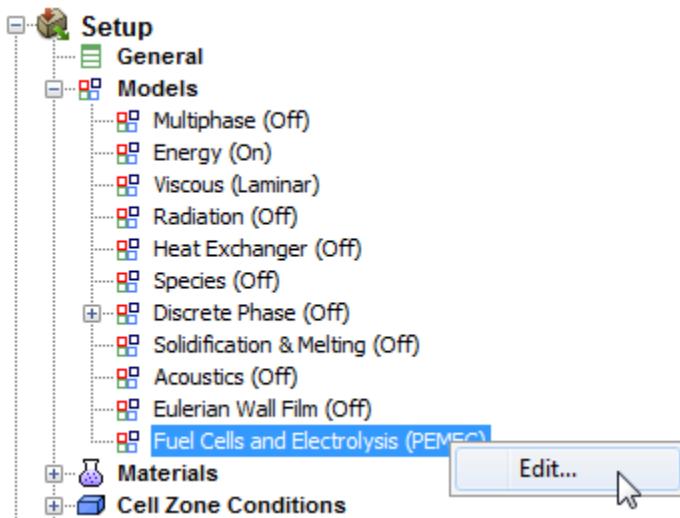
Note that the majority of this chapter describes how to set up the ANSYS Fluent Fuel Cell and Electrolysis Model using the graphical user interface. You can also perform various tasks using the text user interface. For more information, see [Using the Fuel Cell and Electrolysis Text User Interface \(p. 322\)](#).

## **4.6. Modeling Fuel Cells and Electrolysis**

Once the module has been loaded, in order to set fuel cell model parameters and assign properties to the relevant regions in your fuel cell, you need to access the fuel cell graphical user interface (the **Fuel Cell and Electrolysis Models** dialog box).

To open the **Fuel Cell and Electrolysis Models** dialog box: In the tree under the **Models** branch, right-click **Fuel Cells and Electrolysis** and select **Edit...** in the menu that opens.



**Figure 4.1: The Fuel Cell and Electrolysis Option in the Tree**

By default, the PEMFC model is already enabled, however, you can also choose the SOFC or the Electrolysis models.

Using the **Fuel Cell and Electrolysis Models** dialog box, you can identify the relevant zones for the current collectors, flow channels, gas diffusion layers, catalyst layers, and the membrane/electrolyte. You can specify the following inputs using the **Fuel Cell and Electrolysis Models** dialog box. Optional inputs are indicated as such.

1. Enable the appropriate type of fuel cell model, either **PEMFC**, **SOFC**, or **Electrolysis**.
2. Enable either the single-phase or the multi-phase fuel cell model (if **PEMFC** is selected).
3. Set the appropriate options for the fuel cell model (optional).
4. Set the various parameters for the fuel cell model.
5. Select the appropriate zones and specify the properties on the anode side.
6. Select the appropriate zones and specify the properties of the membrane/electrolyte.
7. Select the appropriate zones and specify the properties on the cathode side.
8. Provide input for advanced features such as contact resistivities, coolant channel properties, or stack management settings (optional).
9. Set solution controls such as under-relaxation factors (optional).
10. Provide input to assist reporting (optional).

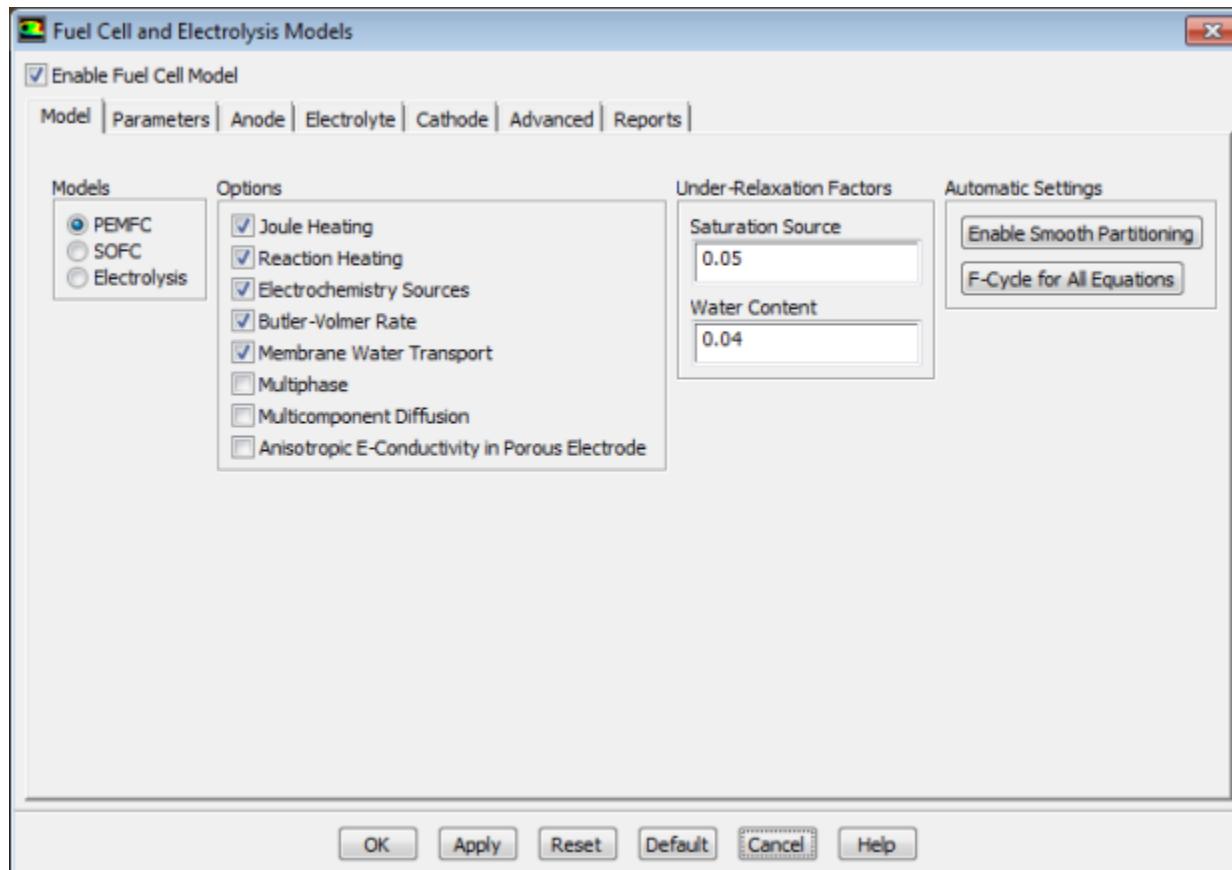
Refer to the following sections for more information:

- [4.6.1. Specifying Model Options](#)
- [4.6.2. Specifying Model Parameters](#)
- [4.6.3. Specifying Anode Properties](#)
- [4.6.4. Specifying Electrolyte/Membrane Properties](#)
- [4.6.5. Specifying Cathode Properties](#)
- [4.6.6. Setting Advanced Properties](#)
- [4.6.7. Reporting on the Solution](#)

## 4.6.1. Specifying Model Options

The **Model** tab of the **Fuel Cell and Electrolysis Models** dialog box allows you to turn on or off various options when solving a fuel cell problem. To model polymer electrolyte membrane fuel cells, enable the **PEMFC** option in the **Model** tab. Likewise, to model solid oxide fuel cells, enable the **SOFc** option in the **Model** tab. Finally, to model electrolysis, enable the **Electrolysis** option in the **Model** tab.

**Figure 4.2: The Model Options in the Fuel Cell and Electrolysis Models Dialog Box—PEMFC Enabled**



Several fuel cell model options are available in the **Model** tab of the **Fuel Cell and Electrolysis Models** dialog box including:

- The **Joule Heating** option takes into account ohmic heating. This option includes the  $I^2R$  term in the energy source term from [Equation 3.25 \(p. 287\)](#) in the calculations.
- The **Reaction Heating** option takes into account the heat generated by the electrochemical reactions, which includes the  $h_{reaction}$  term, and the product of transfer current and the over-potentials in the energy source term from [Equation 3.25 \(p. 287\)](#) in the calculations.
- The **Electrochemistry Sources** option allows the Fuel Cell and Electrolysis Model to take electrochemistry effects into account. If you are only interested in the basic flow field throughout the fuel cell, you can turn off the **Electrochemistry Sources** option in order to suppress most effects of the Fuel Cell and Electrolysis Model. To turn off all effects of the Fuel Cell and Electrolysis Model, you should also turn off the **Membrane Water Transport** and **Multiphase** options.
- The **Butler-Volmer Rate** option (the default) is used to compute the transfer currents inside the catalyst layers. If this option is turned off, the Tafel approximation ([Equation 3.12 \(p. 285\)](#)) is used.

- The **Membrane Water Transport** option takes into account the transport of water across the membrane. This option is only available for the **PEMFC** model.
- The **Multiphase** option takes into account multiphase calculations. Use this option if you are solving for approximate liquid transport in the gas diffusion layer of the fuel cell. (PEMFC only)
- The **Multicomponent Diffusion** option is used to compute the gas species mass diffusivity using the full multicomponent diffusion method as described in [Equation 3.32 \(p. 288\)](#), as opposed to the default option that uses [Equation 3.30 \(p. 288\)](#).
- The **Anisotropic E-Conductivity in Porous Electrode** option is used to model the typically non-isotropic electrical conductivity. It is applicable only for porous electrodes (gas diffusion layers).

Due to the fibrous structure of the porous material that is used for the electrodes (or gas diffusion layer), the electrical conductivity is typically non-isotropical, with the cross-plane components being orders of magnitude smaller than the in-plane components. This can be modeled using the **Anisotropic E-Conductivity in Porous Electrode** setting. When this option is enabled, the **Electrical Conductivity** for the solid material used in the electrolyte is no longer used. Instead, you need to specify, for this solid material, the electrical conductivity by choosing one of the three non-isotropical options for the UDS diffusivity (UDS-0). The three options are: **anisotropic**; **orthotropic**; and **cyl-orthotropic**. For more information about these UDS Diffusivity options, refer to the ANSYS Fluent User's Guide.

For example, to use this feature, perform the following steps:

- Select the **Anisotropic E-Conductivity in Porous Electrode** option in the **Model** tab of the **Fuel Cell and Electrolysis Models** dialog box.
- In the **Create/Edit Materials** dialog box, select **defined-per-uds** for **UDS Diffusivity** for the solid material that is to be used for the porous electrode.
- Select one of the three options for UDS-0: **anisotropic**; **orthotropic**; or **cyl-orthotropic** and set the appropriate values.

---

### Important

Note that, in this case, the **Electrical Conductivity** for *this* solid material is ignored.

---

- For PEMFC problems, you can use the **Under-Relaxation Factors** fields to influence the solution process.

The saturation source term  $r_w$  in [Equation 3.27 \(p. 287\)](#) usually requires under-relaxation. You can change the default value for the under-relaxation factor by changing the value for **Saturation Source**.

The water content,  $\lambda$ , in [Equation 3.38 \(p. 289\)](#) also may need under-relaxation. You can change the default value for the under-relaxation factor by changing the value for **Water Content**.

Under **Automatic Settings**, the following parallel multigrid solver control parameters are available:

- **Enable Smooth Partition:** Enables the smooth partitioning process when running in Parallel.

When this option is enabled, the fuel cell parallel solver:

- Uses the Metis method for partitioning the mesh

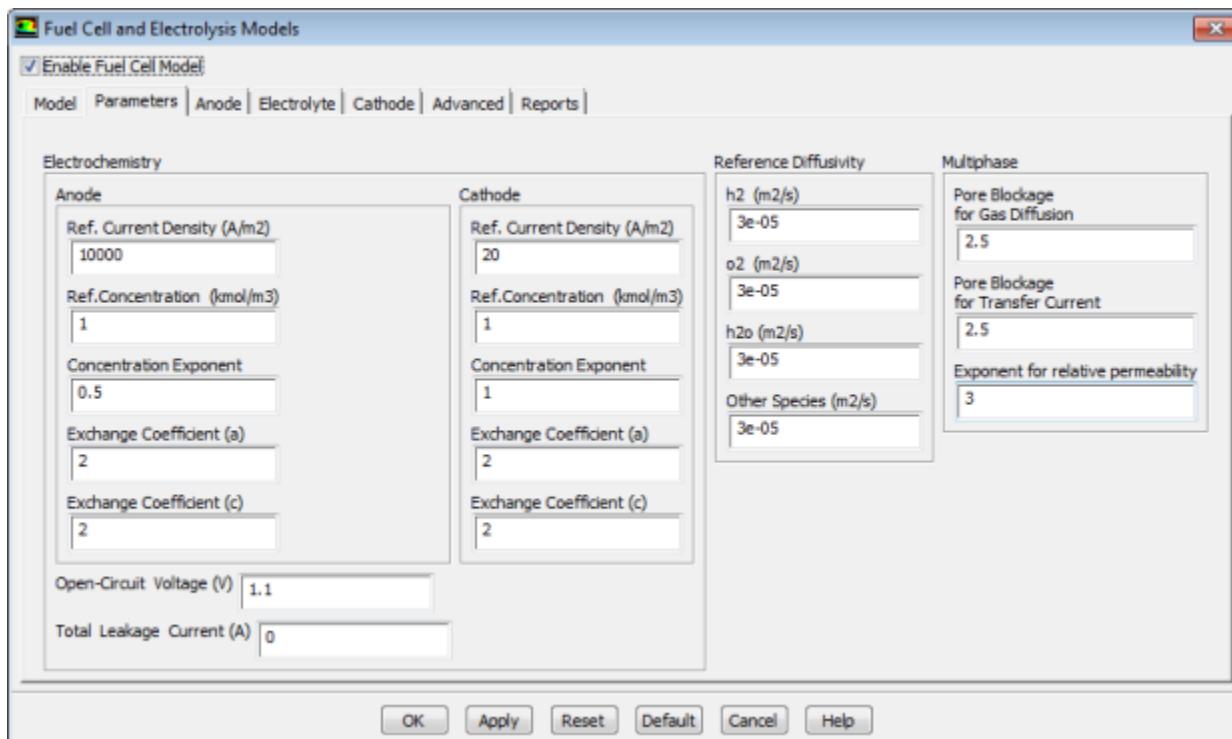
- Enables Laplace Smoothing
- Partitions and distributes the mesh data to all compute nodes
- **F-cycle for All Equations:** Sets the multigrid cycle type to F cycle for all equations that are being solved. This control overrides the equation cycle settings in the **Advanced Solution Controls** dialog box.

Nearly all options are turned on by default. You may want to override the default values, depending on the problem you want to model. For instance, if you are not concerned with the heat generated due to chemical reaction, then you may want to turn off the **Reaction Heating** option.

## 4.6.2. Specifying Model Parameters

You can use the **Parameters** tab of the **Fuel Cell and Electrolysis Models** dialog box to specify the electrochemistry parameters for the Fuel Cell and Electrolysis Model, reference diffusivities for the reactants, among other model parameters.

**Figure 4.3: The Parameters Tab of the Fuel Cell and Electrolysis Models Dialog Box**



There are various parameters under **Electrochemistry** in the **Fuel Cell and Electrolysis Models** dialog box. For both the anode and the cathode, you can also set the following parameters or leave the default values.

- The **Ref. Current Density** corresponds to  $j_{an}^{ref}$  and  $j_{cat}^{ref}$ , the reference exchange current density from [Equation 3.9 \(p. 285\)](#) and [Equation 3.10 \(p. 285\)](#).
- The **Ref. Concentration** corresponds to the reference concentration ( $[H_2]_{ref}$  and  $[O_2]_{ref}$ ) with units of  $kgmol/m^3$  (see [Equation 3.9 \(p. 285\)](#) and [Equation 3.10 \(p. 285\)](#)).
- The **Concentration Exponent** corresponds to  $\gamma$ , the concentration dependence from [Equation 3.9 \(p. 285\)](#).

- The **Exchange Coefficient (a)** and **Exchange Coefficient (c)** correspond to the transfer coefficients,  $\alpha_{an}$  and  $\alpha_{cat}$ , from [Equation 3.9 \(p. 285\)](#) and [Equation 3.10 \(p. 285\)](#).
- The **Open-Circuit Voltage** corresponds to  $V_{oc}$  in [Equation 3.14 \(p. 286\)](#).
- The **Leakage Current** corresponds to  $I_{leak}$  in [Equation 3.42 \(p. 290\)](#) through [Equation 3.47 \(p. 290\)](#). This is the total amount of transfer current (A) due to fuel-oxidant cross-over (leakage through the electrolyte). When this happens, the fuel cell generates less current especially for cases with low values of fuel or air utilization. In addition to the constant value you can specify in the **Electrochemistry** tab, you can also specify the leakage current through the user-defined function `Leakge_Current()`. For more information, see [User-Accessible Functions \(p. 317\)](#).

Moreover, the following parameters can also be set here:

- The **Reference Diffusivities** correspond to  $D_i^0$  from [Equation 3.30 \(p. 288\)](#), the species mass diffusivity. These are not required if the **Multicomponent Diffusion** option is enabled in the **Model** tab.
- The **Pore Blockage for Gas Diffusion** corresponds to  $r_s$  from [Equation 3.30 \(p. 288\)](#) for multiphase PEMFC calculations.
- The **Pore Blockage for Transfer Current** is used to account for liquid blockage to the reaction surface by modifying  $R_{an}$  and  $R_{ca}$  in [Equation 3.9 \(p. 285\)](#) through [Equation 3.12 \(p. 285\)](#) as follows:

$$\begin{aligned} R_{an}^{new} &= (1-s)^r R_{an} \\ R_{cat}^{new} &= (1-s)^r R_{cat} \end{aligned}$$

where  $r$  is the pore blockage for transfer current.

- The **Exponent for relative permeability** corresponds to  $\gamma$  in [Equation 3.28 \(p. 287\)](#).

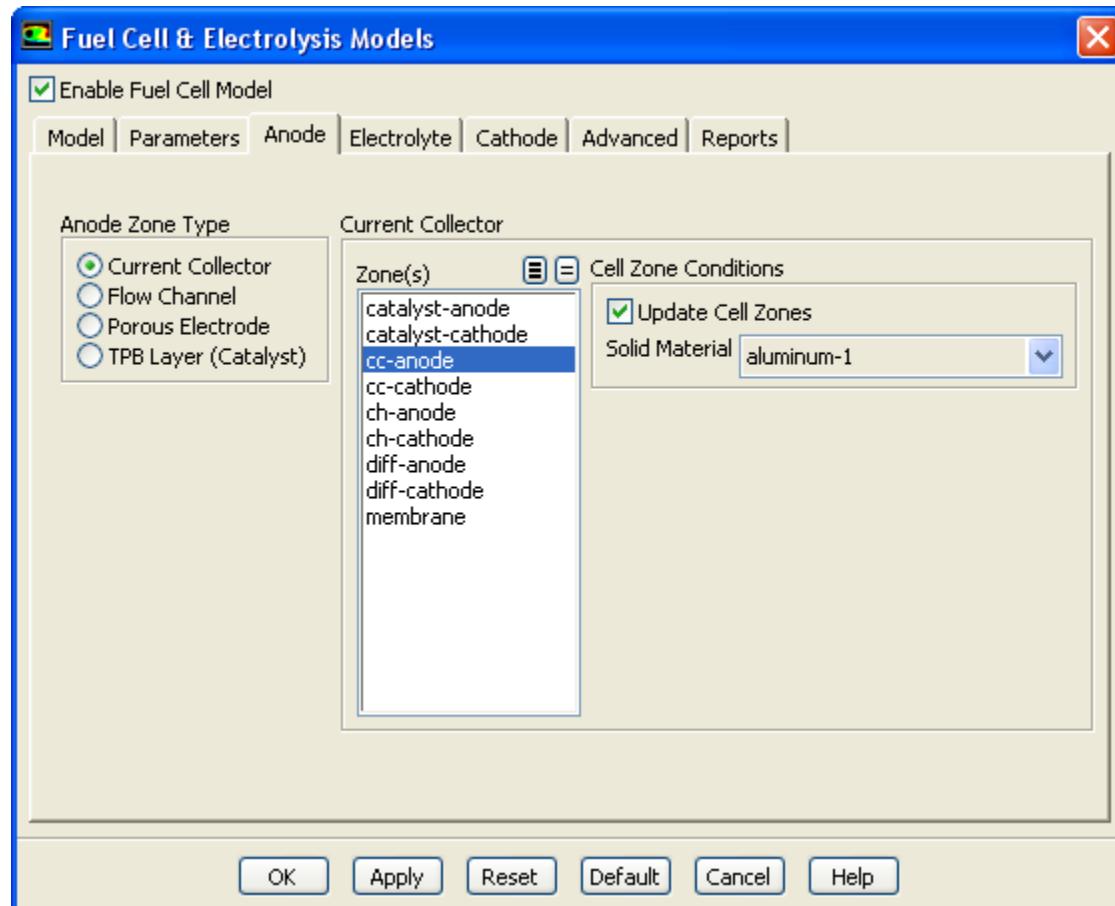
Note that, conventionally, the negative voltage is supplied to the cathode side in power consuming devices such as electrolyzers. ANSYS Fluent adopts the inverse notation where the negative voltage is supplied to the anode side whilst cathode remains positively charged. The main reason for this discrepancy is that the same infrastructure is used for both the electrolysis and fuel cells models. Usage of the terms "anode" and "cathode" in this manual and in the user interface should be interpreted according to conventions for power-supplying devices.

### 4.6.3. Specifying Anode Properties

You can use the **Anode** tab of the **Fuel Cell and Electrolysis Models** dialog box to specify zones and properties of the current collector, the flow channel, the diffusion layer, and the catalyst layer for the anode portion of the fuel cell.

#### 4.6.3.1. Specifying Current Collector Properties for the Anode

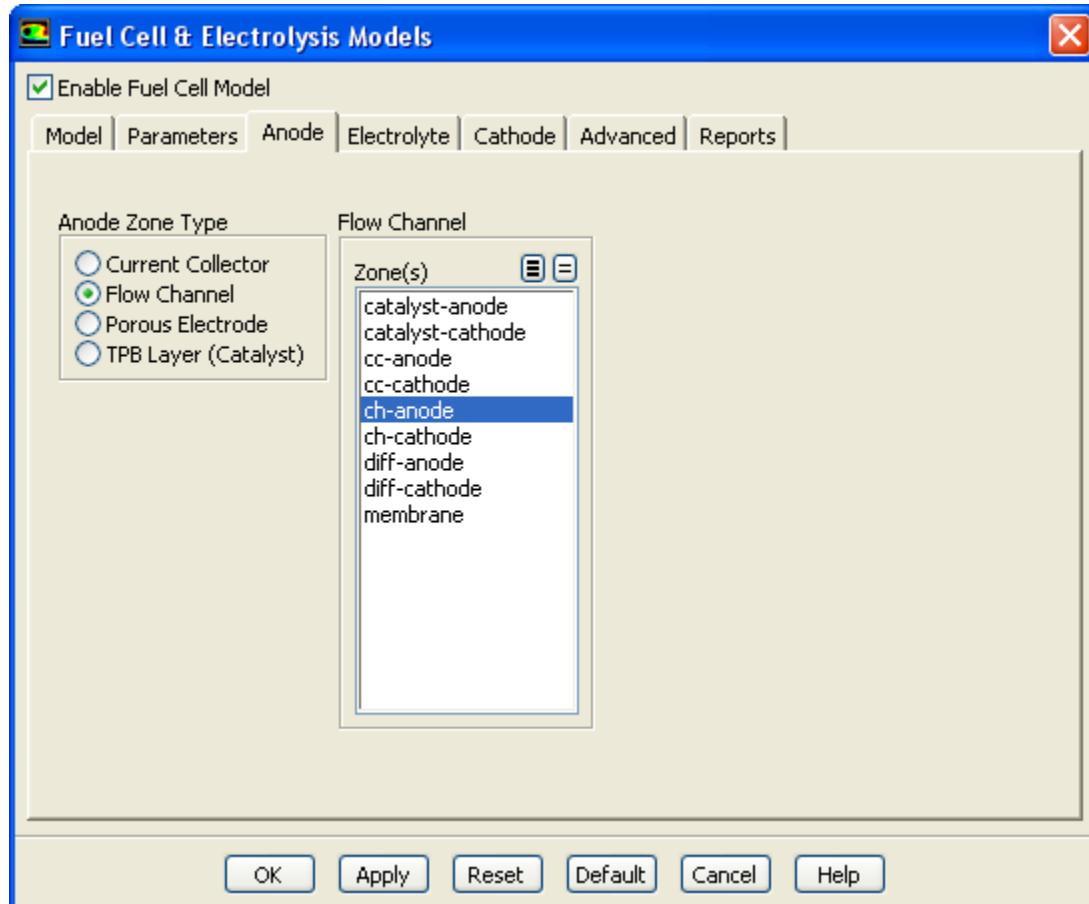
**Figure 4.4: The Anode Tab of the Fuel Cell and Electrolysis Models Dialog Box With Current Collector Selected**



1. Select the **Anode** tab of the **Fuel Cell and Electrolysis Models** dialog box.
2. Select **Current Collector** under **Anode Zone Type**.
3. Select a corresponding zone from the **Zone(s)** list. If you are modeling a fuel cell stack, then you must pick *all* zones of a particular type as a group.
4. Select a **Solid Material** from the corresponding drop-down list. Solid materials can be customized using the **Create/Edit Materials** dialog box. Note that for the **Electrical Conductivity**, you can only choose a constant value in the **Create/Edit Materials** dialog box. The solid electrical conductivity value is the diffusivity of the solid phase potential in the solid zones.

#### 4.6.3.2. Specifying Flow Channel Properties for the Anode

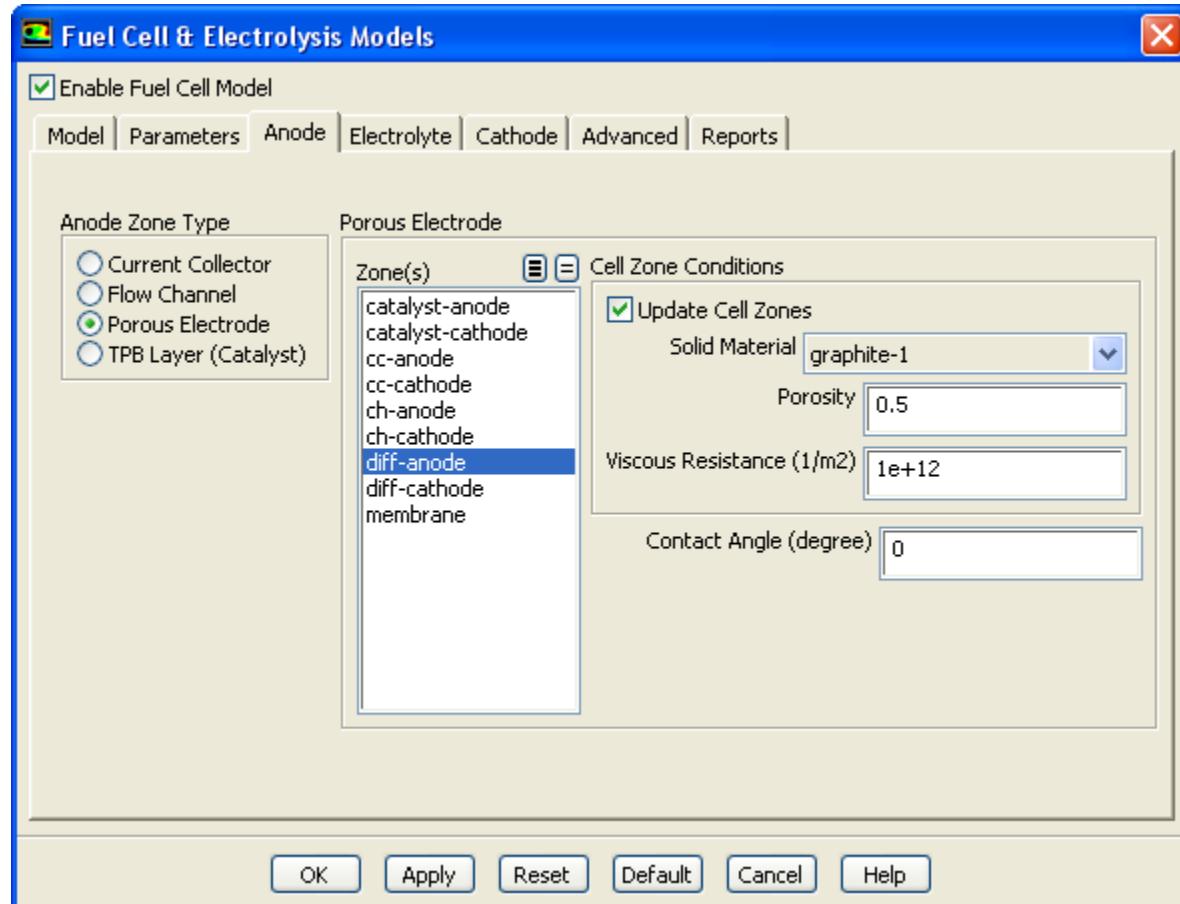
Figure 4.5: The Anode Tab of the Fuel Cell and Electrolysis Models Dialog Box With Flow Channel Selected



1. Select the **Anode** tab of the **Fuel Cell and Electrolysis Models** dialog box.
2. Select **Flow Channel** under **Anode Zone Type**.
3. Select a corresponding zone from the **Zone(s)** list.

### 4.6.3.3. Specifying Porous Electrode Properties for the Anode

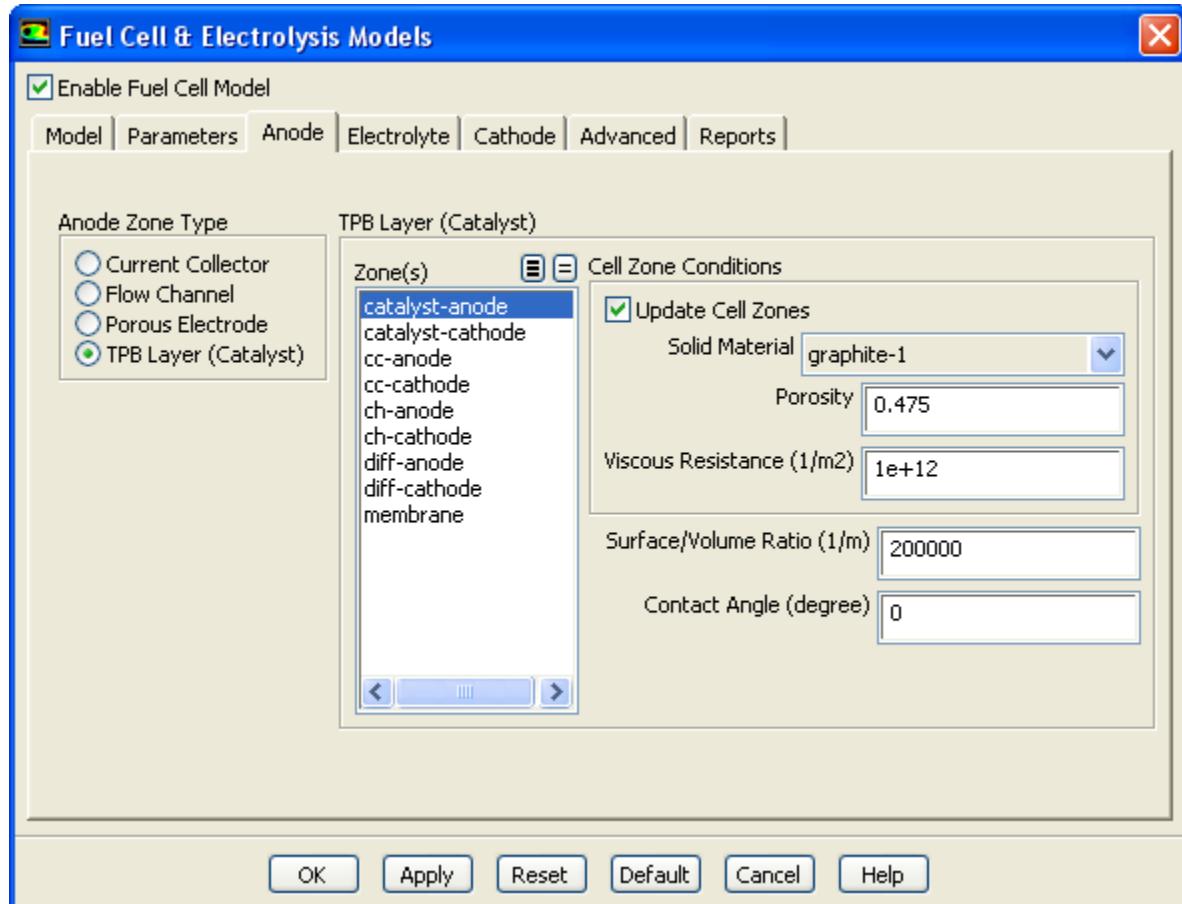
**Figure 4.6: The Anode Tab of the Fuel Cell and Electrolysis Models Dialog Box With Porous Electrode Selected**



1. Select the **Anode** tab of the **Fuel Cell and Electrolysis Models** dialog box.
2. Select **Porous Electrode** under **Anode Zone Type**.
3. Select a corresponding zone from the **Zone(s)** list. If you are modeling a fuel cell stack, then you must pick *all* zones of a particular type as a group.
4. Select a **Solid Material** from the corresponding drop-down list. Solid materials can be customized using the **Create/Edit Materials** dialog box. Note that for the **Electrical Conductivity**, you can only choose a constant value in the **Create/Edit Materials** dialog box. The solid electrical conductivity value is the diffusivity of the solid phase potential in the solid zones.
5. Specify a value for the **Porosity**.
6. Specify a value for the **Viscous Resistance**.
7. Specify a value for the **Contact Angle** for multiphase fuel cell calculations ( $\theta_c$  in [Equation 3.29 \(p. 287\)](#)).

#### 4.6.3.4. Specifying Catalyst Layer Properties for the Anode

**Figure 4.7: The Anode Tab of the Fuel Cell and Electrolysis Models Dialog Box With TPB Layer (Catalyst) Selected**



1. Select the **Anode** tab of the **Fuel Cell and Electrolysis Models** dialog box.
2. Select **TPB Layer (Catalyst)** under **Anode Zone Type**.
3. Select a corresponding zone from the **Zone(s)** list. If you are modeling a fuel cell stack, then you must pick *all* zones of a particular type as a group.
4. Select a **Solid Material** from the corresponding drop-down list. Solid materials can be customized using the **Create/Edit Materials** dialog box. Note that for the **Electrical Conductivity**, you can only choose a constant value in the **Create/Edit Materials** dialog box. The solid electrical conductivity value is the diffusivity of the solid phase potential in the solid zones.
5. Specify a value for the **Porosity**.
6. Specify a value for the **Viscous Resistance**.
7. Specify a value for the **Surface/Volume Ratio** (the specific active surface area in [Equation 3.9 \(p. 285\)](#)).
8. Specify a value for the **Contact Angle** for multiphase fuel cell calculations ( $\theta_c$  in [Equation 3.29 \(p. 287\)](#)).

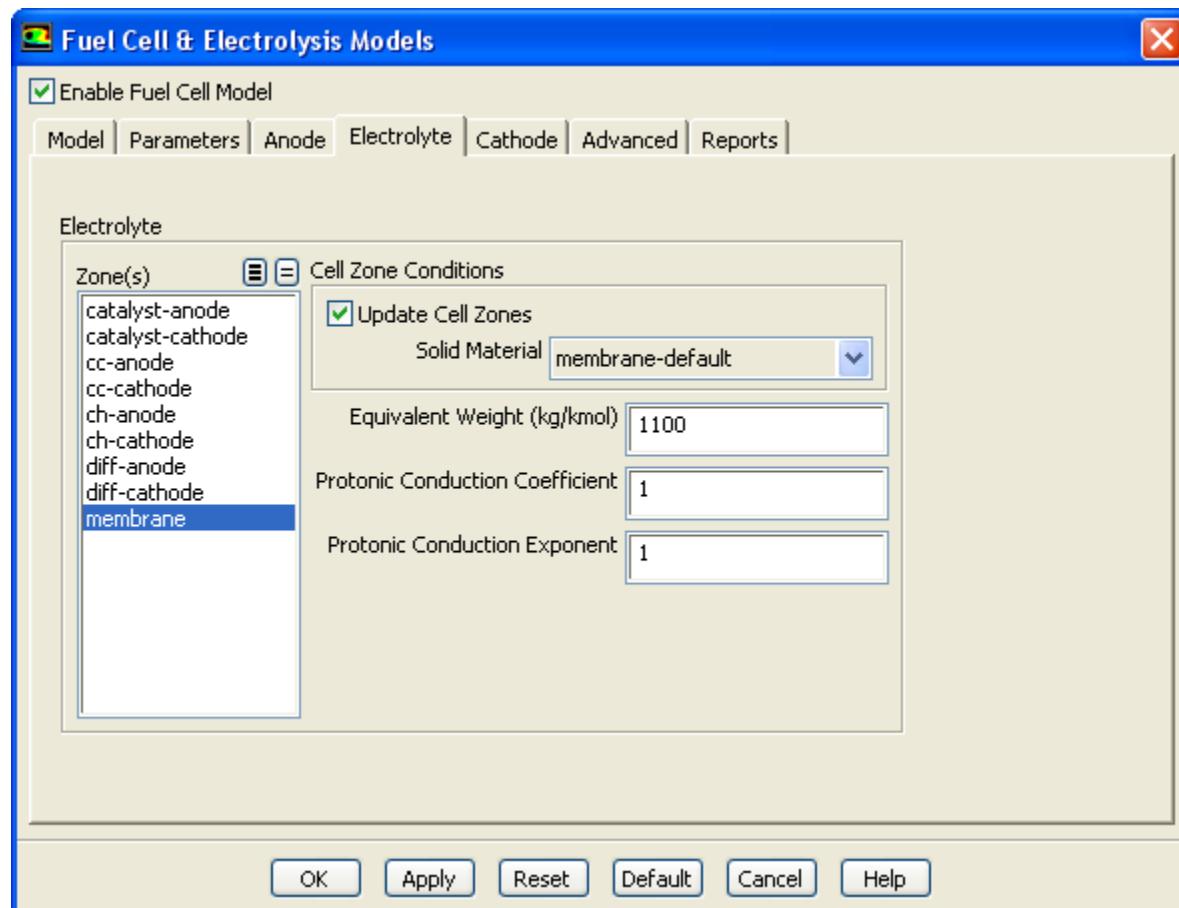
### 4.6.3.5. Specifying Cell Zone Conditions for the Anode

For each case of the anode's current collector, diffusion layer, and catalyst layer, you assign a solid material and/or set the porosity and the viscous resistance. These settings represent setting a cell zone condition. With the **Update Cell Zones** option turned on (the default setting), this cell zone condition is applied to all selected zones in the **Zone(s)** list. If you want to set the cell zone conditions for each zone individually (using the **Cell Zone Conditions** task page), you should turn off the **Update Cell Zones** option.

### 4.6.4. Specifying Electrolyte/Membrane Properties

You can use the **Electrolyte** tab of the **Fuel Cell and Electrolysis Models** dialog box to specify zones and properties of the electrolyte/membrane portion of the fuel cell.

**Figure 4.8: The Electrolyte Tab of the Fuel Cell and Electrolysis Models Dialog Box**



1. Select a corresponding zone from the **Zone(s)** list. If you are modeling a fuel cell stack, then you must pick *all* membrane zones as a group.
2. Select a **Solid Material** from the corresponding drop-down list. Solid materials can be customized using the **Create/Edit Materials** dialog box.
3. Specify a value for the **Equivalent Weight** ( $M_m$  in [Equation 3.36 \(p. 289\)](#)).
4. Specify a value for the **Protonic Conduction Coefficient** ( $\beta$  in [Equation 3.34 \(p. 289\)](#)). This is used to calculate the membrane phase electric conductivity.

## 5. Specify a value for the **Protonic Conduction Exponent** ( $\omega$ in [Equation 3.34 \(p. 289\)](#)).

Note that the Fuel Cell and Electrolysis Model allows you to model the electrolyte/membrane as either a fluid zone or as a solid zone. For PEMFC, the Fuel Cell and Electrolysis Model still allows for water and the ionic current to pass through the electrolyte/membrane.

### 4.6.4.1. Specifying Cell Zone Conditions for the Membrane

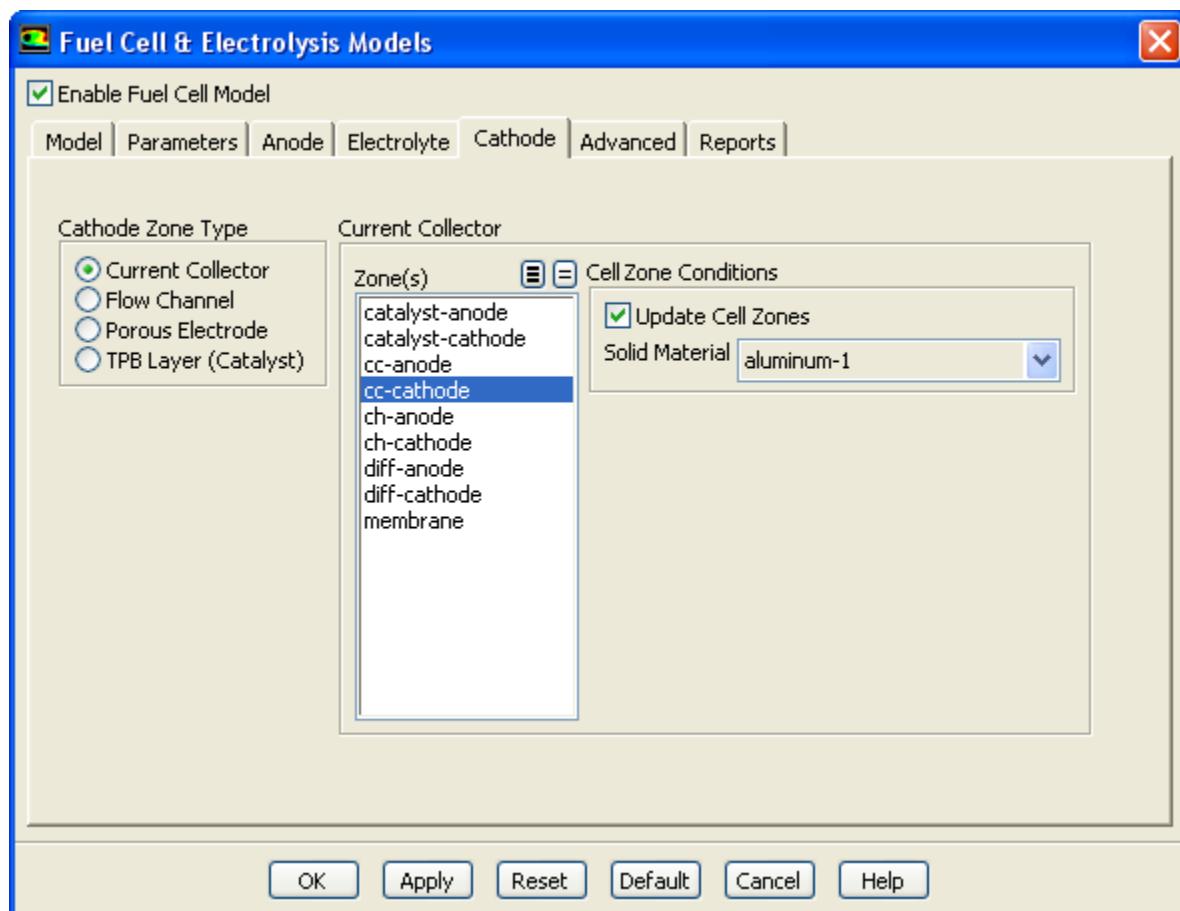
When you assign a solid material to the membrane, you are setting a cell zone condition. With the **Update Cell Zones** option turned on (the default setting), this cell zone condition is applied to all selected zones in the **Zone(s)** list. If you want to set the cell zone conditions for each zone individually (using the **Cell Zone Conditions** task page), you should turn off the **Update Cell Zones** option.

## 4.6.5. Specifying Cathode Properties

You can use the **Cathode** tab of the **Fuel Cell and Electrolysis Models** dialog box to specify zones and properties of the current collector, the flow channel, the diffusion layer, and the catalyst layer for the cathode portion of the fuel cell.

### 4.6.5.1. Specifying Current Collector Properties for the Cathode

**Figure 4.9: The Cathode Tab of the Fuel Cell and Electrolysis Models Dialog Box With Current Collector Selected**

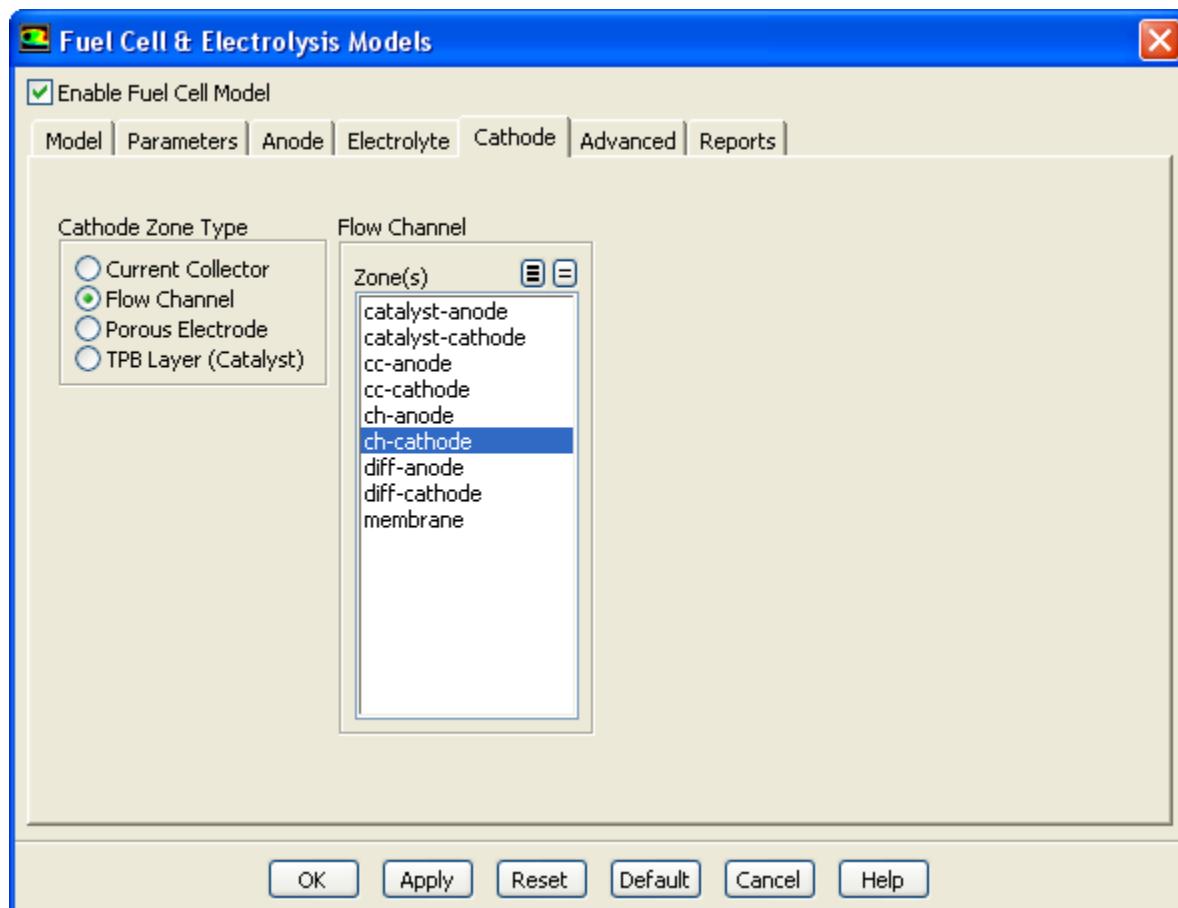


- Select the **Cathode** tab of the **Fuel Cell and Electrolysis Models** dialog box.

2. Select **Current Collector** under **Cathode Zone Type**.
3. Select a corresponding zone from the **Zone(s)** list. If you are modeling a fuel cell stack, then you must pick *all* zones of a particular type as a group.
4. Select a **Solid Material** from the corresponding drop-down list. Solid materials can be customized using the **Create/Edit Materials** dialog box. Note that for the **Electrical Conductivity**, you can only choose a constant value in the **Create/Edit Materials** dialog box. The solid electrical conductivity value is the diffusivity of the solid phase potential in the solid zones.

#### **4.6.5.2. Specifying Flow Channel Properties for the Cathode**

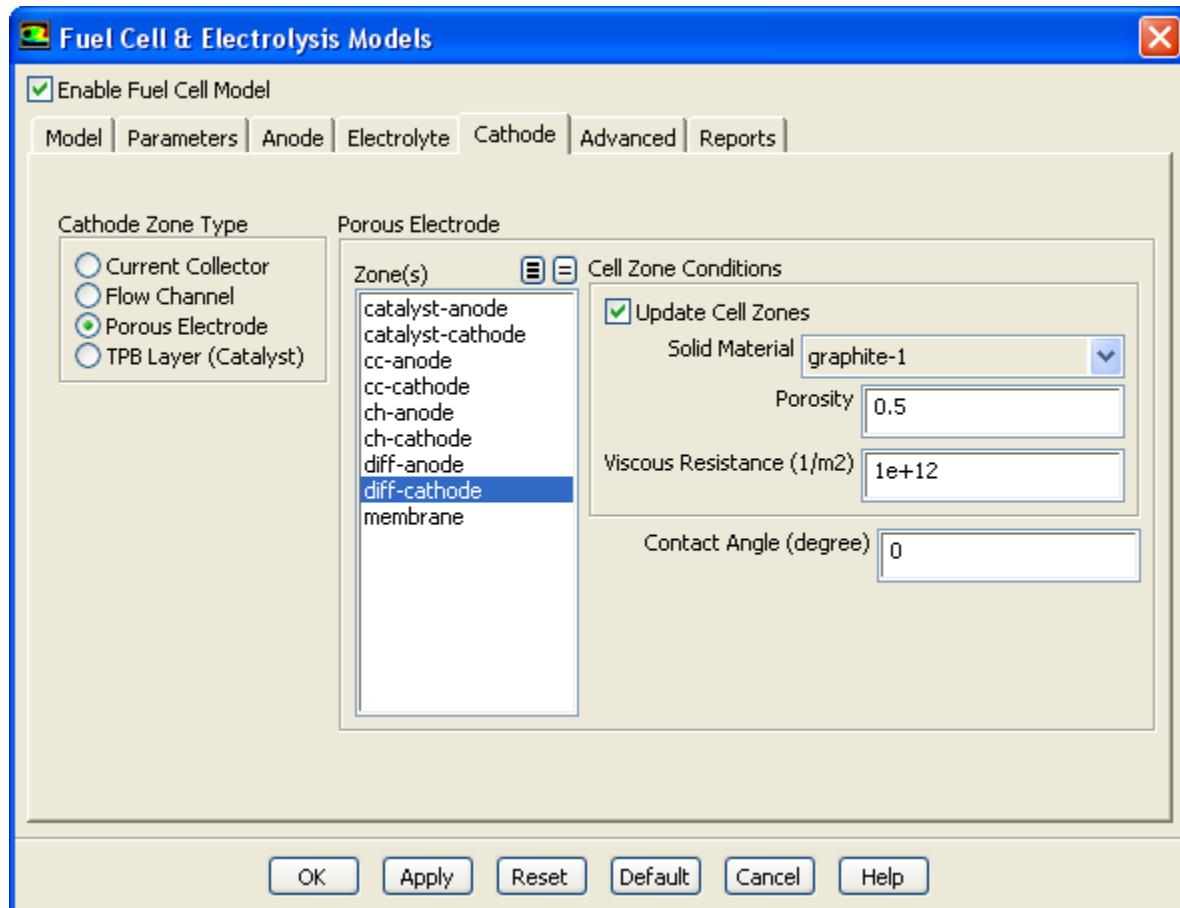
**Figure 4.10: The Cathode Tab of the Fuel Cell and Electrolysis Models Dialog Box With Flow Channel Selected**



1. Select the **Cathode** tab of the **Fuel Cell and Electrolysis Models** dialog box.
2. Select **Flow Channel** under **Cathode Zone Type**.
3. Select a corresponding zone from the **Zone(s)** list. If you are modeling a fuel cell stack, then you must pick *all* zones of a particular type as a group.

#### 4.6.5.3. Specifying Porous Electrode Properties for the Cathode

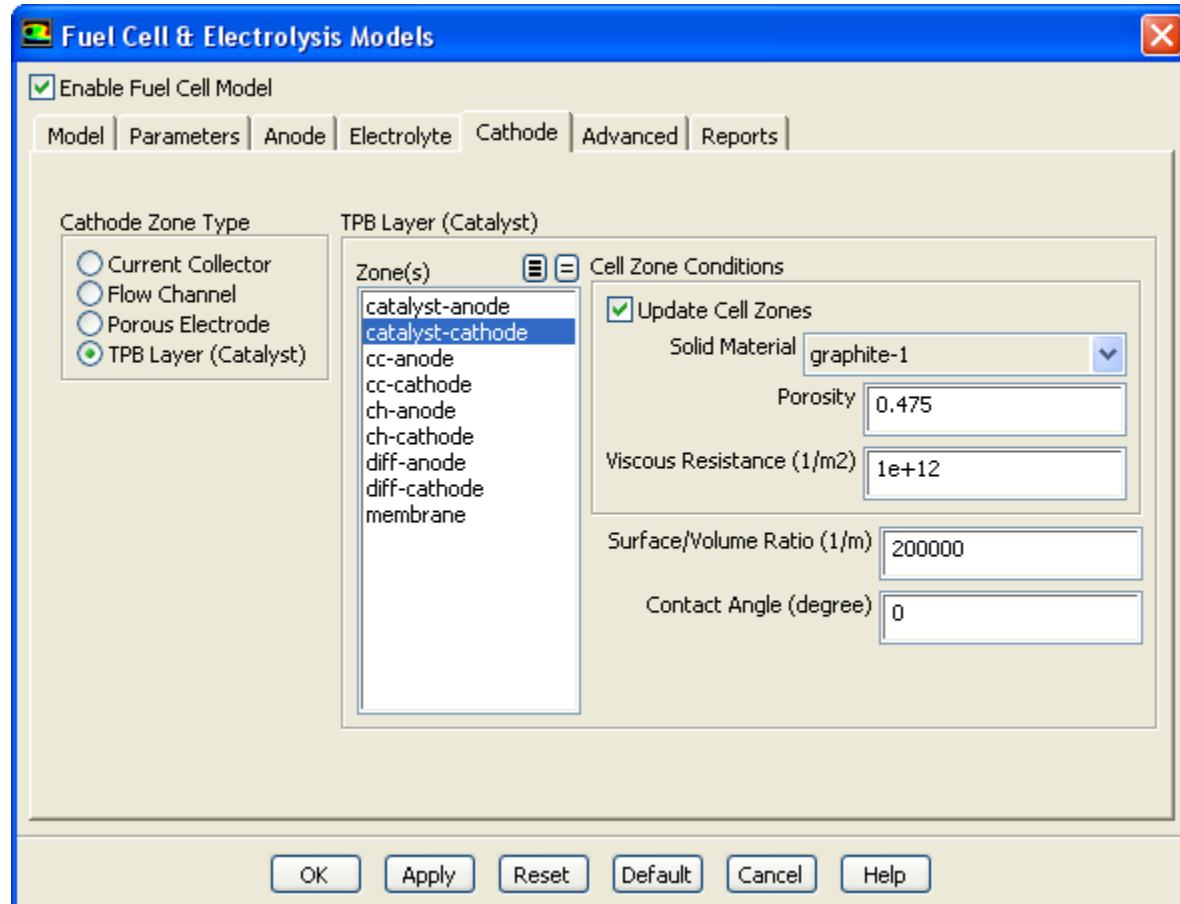
**Figure 4.11: The Cathode Tab of the Fuel Cell and Electrolysis Models Dialog Box With Porous Electrode Selected**



1. Select the **Cathode** tab of the **Fuel Cell and Electrolysis Models** dialog box.
2. Select **Porous Electrode** under **Cathode Zone Type**.
3. Select a corresponding zone from the **Zone(s)** list. If you are modeling a fuel cell stack, then you must pick *all* zones of a particular type as a group.
4. Select a **Solid Material** from the corresponding drop-down list. Solid materials can be customized using the **Create/Edit Materials** dialog box. Note that for the **Electrical Conductivity**, you can only choose a constant value in the **Create/Edit Materials** dialog box. The solid electrical conductivity value is the diffusivity of the solid phase potential in the solid zones.
5. Specify a value for the **Porosity**.
6. Specify a value for the **Viscous Resistance**.
7. Specify a value for the **Contact Angle** for multiphase fuel cell calculations ( $\theta_c$  in [Equation 3.29 \(p. 287\)](#)).

#### 4.6.5.4. Specifying Catalyst Layer Properties for the Cathode

**Figure 4.12: The Cathode Tab of the Fuel Cell and Electrolysis Models Dialog Box With TPB Layer (Catalyst) Selected**



1. Select the **Cathode** tab of the **Fuel Cell and Electrolysis Models** dialog box.
2. Select **TPB Layer(Catalyst)** under **Cathode Zone Type**.
3. Select a corresponding zone from the **Zone(s)** list. If you are modeling a fuel cell stack, then you must pick *all* zones of a particular type as a group.
4. Select a **Solid Material** from the corresponding drop-down list. Solid materials can be customized using the **Create/Edit Materials** dialog box. Note that for the **Electrical Conductivity**, you can only choose a constant value in the **Create/Edit Materials** dialog box. The solid electrical conductivity value is the diffusivity of the solid phase potential in the solid zones.
5. Specify a value for the **Porosity**.
6. Specify a value for the **Viscous Resistance**.
7. Specify a value for the **Surface/Volume Ratio** (the specific active surface area in [Equation 3.10 \(p. 285\)](#)).
8. Specify a value for the **Contact Angle** for multiphase fuel cell calculations ( $\theta_c$  in [Equation 3.29 \(p. 287\)](#)).

#### 4.6.5.5. Specifying Cell Zone Conditions for the Cathode

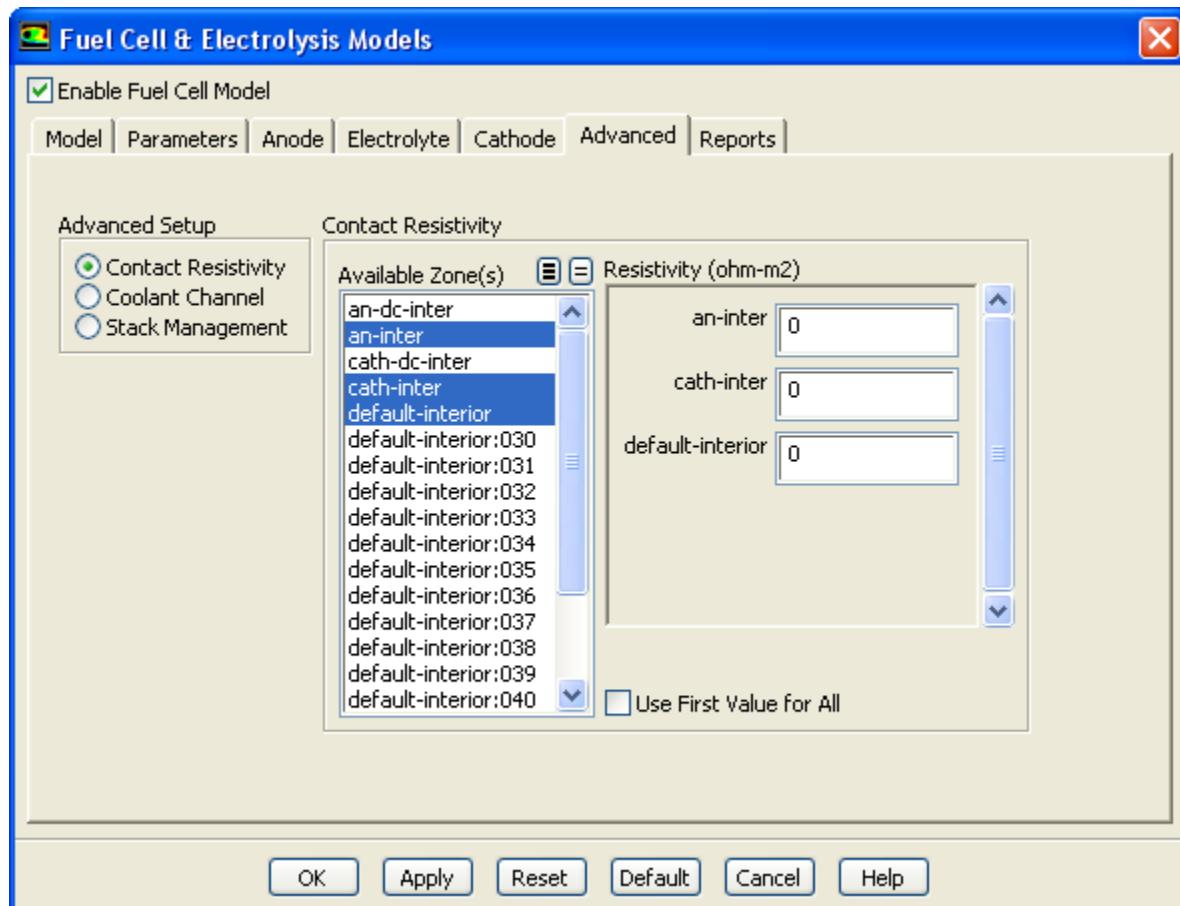
For each case of the cathode's current collector, diffusion layer, and catalyst layer, you assign a solid material and/or set the porosity and the viscous resistance. These settings represent setting a cell zone condition. With the **Update Cell Zones** option turned on (the default setting), this cell zone condition is applied to all selected zones in the **Zone(s)** list. If you want to set the cell zone conditions for each zone individually (using the **Cell Zone Conditions** task page), you should turn off the **Update Cell Zones** option.

#### 4.6.6. Setting Advanced Properties

You can use the **Advanced** tab of the **Fuel Cell and Electrolysis Models** dialog box to specify the contact resistivity for any material interface in the geometry, set parameters for coolant channels, and define fuel stack units for managing stacks of fuel cells.

##### 4.6.6.1. Setting Contact Resistivities for the Fuel Cell and Electrolysis Model

**Figure 4.13: The Advanced Tab of the Fuel Cell and Electrolysis Models Dialog Box for Contact Resistivities**

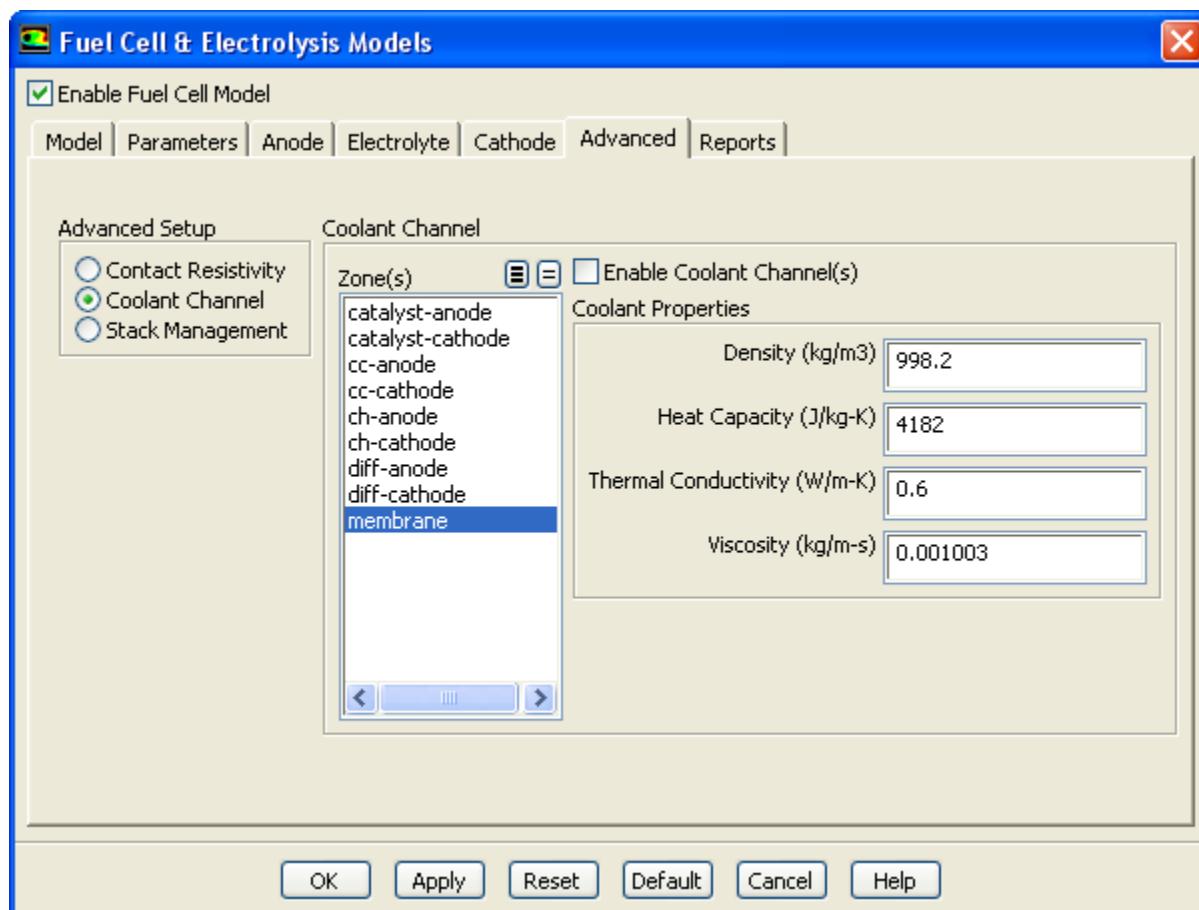


1. Select the **Advanced** tab of the **Fuel Cell and Electrolysis Models** dialog box.
2. Select **Contact Resistivity** under **Advanced Setup**.
3. Select any number of corresponding interfaces from the **Available Zone(s)** list. These zones are face zones over which a jump in electrical potential is caused by imperfect conduction.

4. Specify a value for the **Resistivity** for each specified zone.
5. To simplify the input, you can choose to use the resistivity value of the first selected zone for all others as well by turning on the **Use First Value for All** option.

#### **4.6.6.2. Setting Coolant Channel Properties for the Fuel Cell and Electrolysis Model**

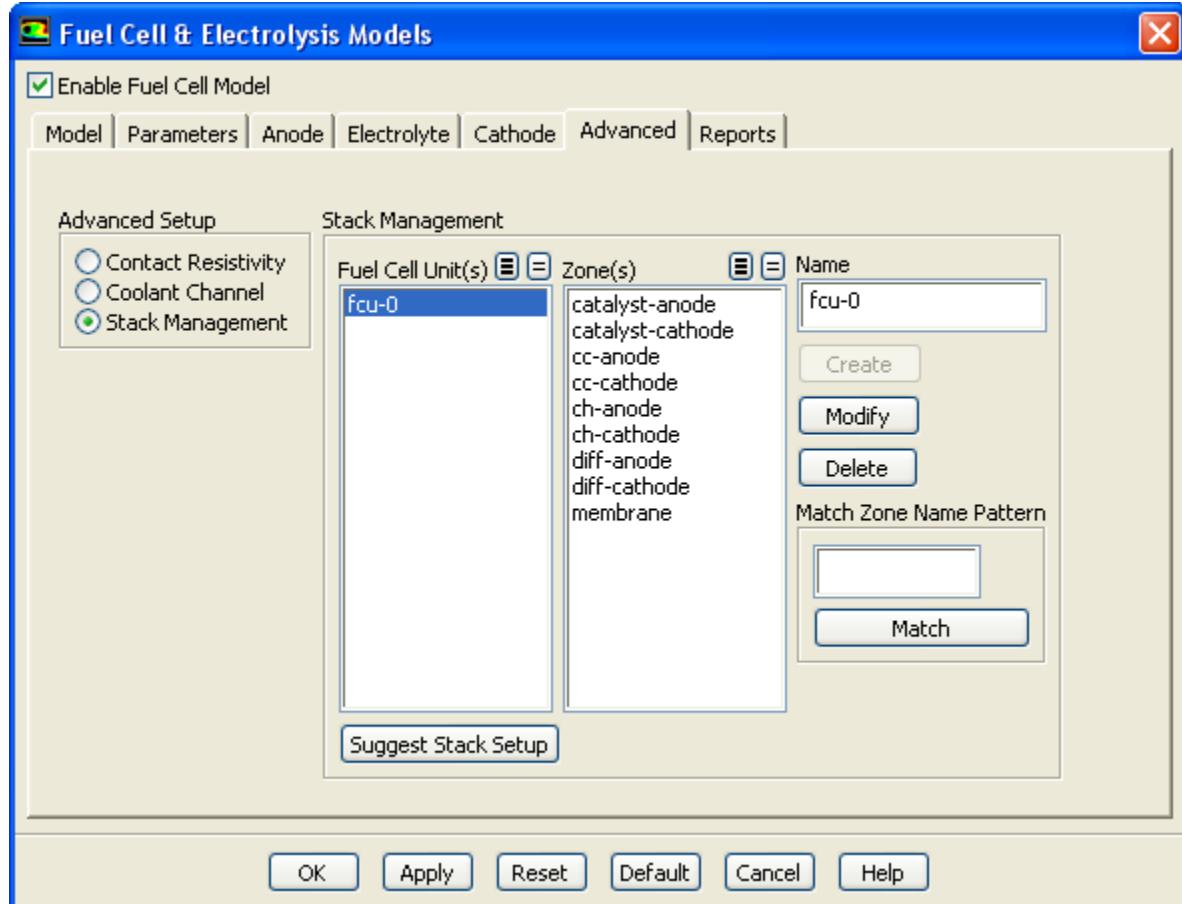
**Figure 4.14: The Advanced Tab of the Fuel Cell and Electrolysis Models Dialog Box for the Coolant Channel**



1. Select the **Advanced** tab of the **Fuel Cell and Electrolysis Models** dialog box.
2. Select **Coolant Channel** under **Advanced Setup**.
3. Select any number of corresponding zones from the **Zone(s)** list.
4. Specify a value for the **Density**.
5. Specify a value for the **Heat Capacity**.
6. Specify a value for the **Thermal Conductivity**.
7. Specify a value for the **Viscosity**.
8. To enable the coolant channel, turn on the **Enable Coolant Channel(s)** option. Amongst other settings, this will change the mixture to include the coolant species, which is otherwise absent.

#### 4.6.6.3. Managing Stacks for the Fuel Cell and Electrolysis Model

**Figure 4.15: The Advanced Tab of the Fuel Cell and Electrolysis Models Dialog Box for Stack Management**



The ANSYS Fluent Fuel Cell and Electrolysis Model allows you to model fuel cell stacks as well as individual fuel cells. In the **Advanced** tab of the **Fuel Cell and Electrolysis Models** dialog box, you can define *fuel cell units* for each fuel cell in a stack. A fuel cell unit consists of all zones of a single fuel cell in the stack.

#### Important

If you are only modeling a single fuel cell, then you do not need to set anything for **Stack Management** in the **Advanced** tab of the **Fuel Cell and Electrolysis Models** dialog box.

1. Select the **Advanced** tab of the **Fuel Cell and Electrolysis Models** dialog box.
2. Select **Stack Management** under **Advanced Setup**.
3. Since a fuel cell unit consists of all zones of a single fuel cell in the stack, select the corresponding zones from the **Zone(s)** list.
4. Create a new fuel cell unit by clicking the **Create** button. The new fuel cell is listed under **Fuel Cell Unit(s)** with a default name.

5. Edit a pre-existing fuel cell unit by selecting it in the **Fuel Cell Unit(s)** list. The zones in this fuel cell unit are automatically selected in the **Zone(s)** list. You can then modify the zones that comprise the fuel cell unit and/or change its name in the **Name** field and click **Modify** to save the new settings.
6. Remove a pre-existing fuel cell unit by selecting it in the **Fuel Cell Unit(s)** list and clicking the **Delete** button.
7. If your model contains many zone names, you can use the **Match Zone Name Pattern** field to specify a pattern to look for in the names of zones. Type the pattern in the text field and click **Match** to select (or deselect) the zones in the **Zones** list with names that match the specified pattern. You can match additional characters using \* and ?. For example, if you specify wall\*, all surfaces whose names begin with **wall** (for example, **wall-1**, **wall-top**) will be selected automatically. If they are all selected already, they will be deselected. If you specify wall?, all surfaces whose names consist of **wall** followed by a single character will be selected (or deselected, if they are all selected already).

For example, in a stack there are many fuel cells, say 10–100, each having at least 9 zones (current collector, gas channel, diffusion layer, and catalyst layer for both anode and cathode and a membrane). Additionally, there may be coolant channels, and it may be that for mesh construction reasons each of these physical zones is made up of more than one mesh zone. Even for small stacks, you can easily end up having hundreds of cell zones in an ANSYS Fluent mesh. Therefore, you may want to consider numbering the fuel cells in a stack and to use the assigned fuel cell number in the names of the mesh zones. When you set up your stacked fuel cell case, you would use the **Match Zone Name Pattern** field to pick all the zones belonging to a single fuel cell in the stack, rather than scrolling through the potentially very long list and selecting them manually.

8. You can have ANSYS Fluent attempt to automatically determine the zones that constitute a single fuel cell in a stack using the **Suggest Stack Setup** button. Manually performing this task is often time-consuming and error-prone. Using the **Suggest Stack Setup** button can save you from having to manually enter this information yourself for potentially hundreds of zones.

When using the **Suggest Stack Setup** button, ANSYS Fluent needs to correctly identify electrically conducting parts and their connectivity (anode, electrolyte, cathode, coolant channels, and external contacts) using zone information generally required by the fuel cell model anyway. ANSYS Fluent requires zone information to have been specified in *all* of the following tabs in the **Fuel Cell and Electrolysis Models** dialog box:

- In the **Anode** tab, specify zone information for the current collector, the porous electrode, and the TPB catalyst layer.
- In the **Electrolyte** tab, specify zone information for the electrolyte/membrane.
- In the **Cathode** tab, specify zone information for the current collector, the porous electrode, and the TPB catalyst layer.
- In the **Advanced** tab, specify zone information for the coolant channel.
- In the **Reports** tab, specify the external contact interface(s).

- In the **Advanced** tab, click the **Suggest Stack Setup** button.

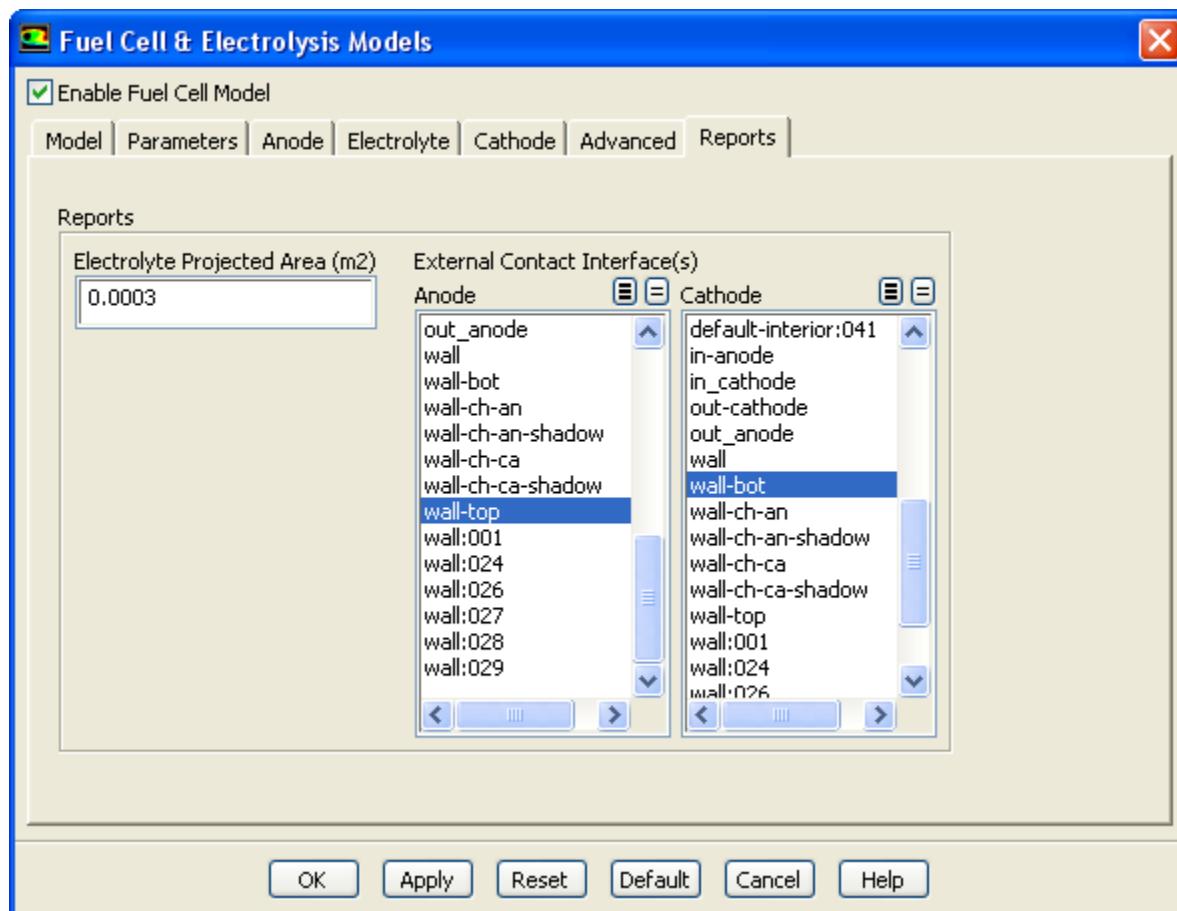
### Important

If your fuel cell model inputs are incorrect (or incomplete), ANSYS Fluent cannot completely be sure that the resulting setup is correct. Also, even if your inputs are correct, an unconventional fuel cell design may cause ANSYS Fluent to suggest an inaccurate stack setup. So, it is your responsibility to verify the stack setup prior to clicking either the **OK** or the **Apply** buttons.

## 4.6.7. Reporting on the Solution

You can use the **Reports** tab of the **Fuel Cell and Electrolysis Models** dialog box to set up parameters that will be useful in reporting data relevant to the fuel cell.

**Figure 4.16: The Reports Tab of the Fuel Cell and Electrolysis Models Dialog Box**



The **Electrolyte Projected Area** field requires the projected area of the Membrane Electrolyte Assembly (MEA) and is only used to calculate the average current density. The assembly consists of the membrane and the catalyst layers above and below the membrane. The value of the projected area can be computed from the **Projected Surface Areas** dialog box.

**Results** → **Reports** → **Projected Areas** **Edit...**

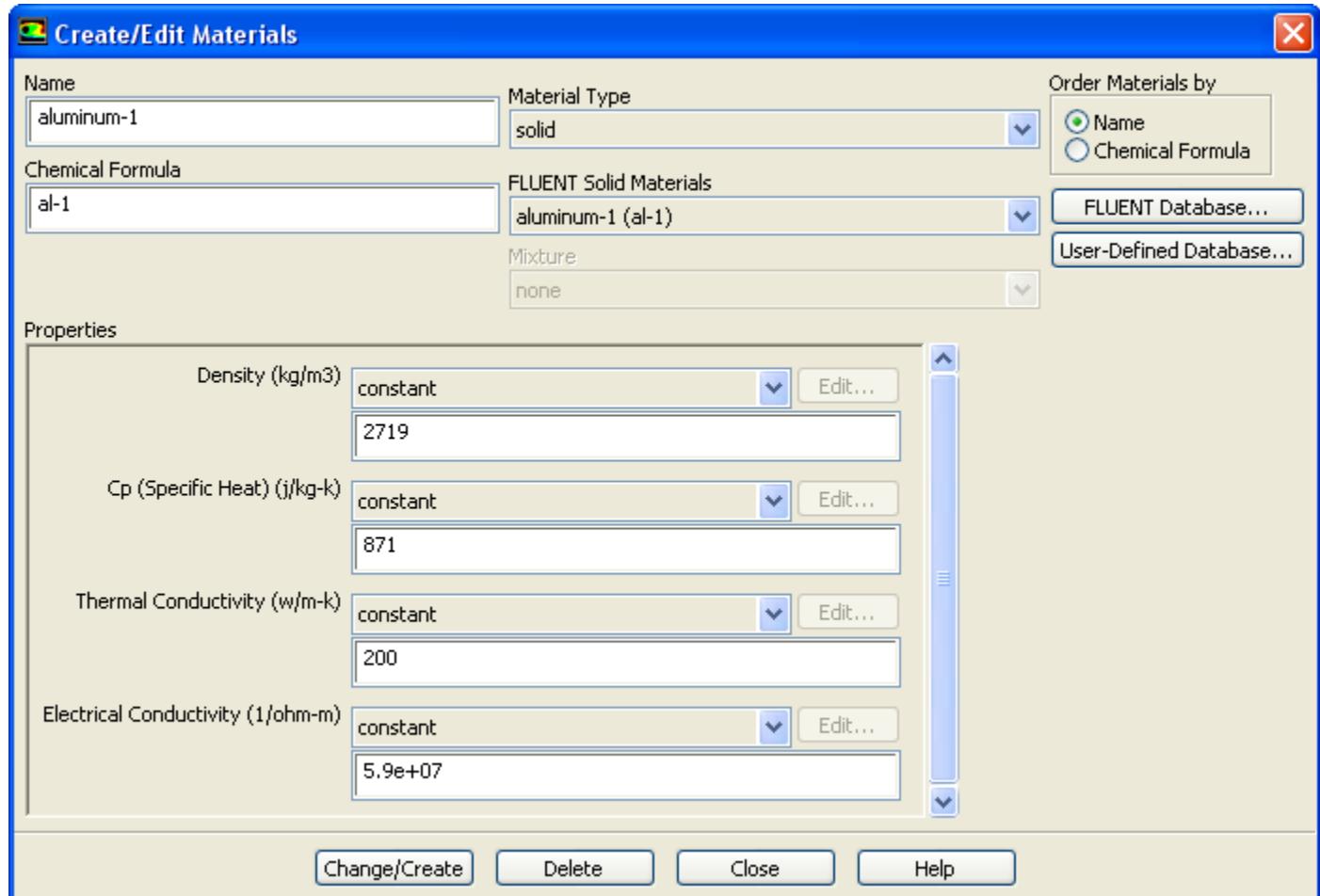
The **External Contact Interface(s)** fields requires the face zones that act as external contact surfaces for the anode and the cathode.

These inputs are used to report cell voltage. For potentiostatic boundary conditions, this is the difference between the provided values, but for galvanostatic boundary conditions, the cell voltage is part of the solution.

## 4.7. Modeling Current Collectors

In previous versions of ANSYS Fluent, user-defined scalar (UDS) equations could only be solved in fluid zones. This restriction is now removed. As a result, the Fuel Cell and Electrolysis module allows you to model current collectors as solid, as well as fluid zones. One advantage of using solids as the current collector is that the convergence of the species equations are not hindered by the potentially skewed mesh inside the current collectors.

If fluid zones are used to model solid current collectors, ANSYS Fluent automatically sets velocities to zero and cuts off species transport into these zones. If solid zones are used, however, you need to activate the solution of the electric potential (UDS-0) in these solid zones (see the separate ANSYS Fluent User's Guide for details). The value of the **Electric Conductivity** for the solid material must be assigned in the **Create/Edit Materials** dialog box.

**Figure 4.17: The Electric Conductivity Field in the Create/Edit Materials Dialog Box**

### Important

Note that the **UDS Diffusivity** should be set to user-defined (cond::fuelcells). Do not use the defined-per-uds option.

For more information on the user-defined scalar diffusivity, see the separate ANSYS Fluent [User's Guide](#).

Note that the Fuel Cell and Electrolysis Model allows you to model current collectors either as porous media zones (if you want to allow for mass and momentum transport within the collectors) or as solid zones.

## 4.8. Fuel Cell and Electrolysis Model Boundary Conditions

The following boundary conditions need to be defined for the Fuel Cell and Electrolysis simulation based on your problem specification:

- Anode Inlet
  - Mass flow rate
  - Temperature

- Direction specification method
- Mass fractions (for example, **h2**, and **h2o**).
- The coolant *must* be set to zero if coolant channels are enabled.
- UDS-2 (Water Saturation) must be set to 0
- Cathode Inlet
  - Mass flow rate
  - Temperature
  - Direction specification method
  - Mass fractions (for example **o2**, **h2o**, and **n2**).
  - The coolant *must* be set to zero if coolant channels are enabled.
  - UDS-2 (Water Saturation) must be set to 0
- Coolant Inlet (if any)
  - Mass flow rate
  - Temperature
  - Direction specification method
  - Coolant mass fraction set to 1
  - UDS-2 (Water Saturation) must be set to 0
- Pressure Outlets (all)
 

Realistic backflow conditions.
- Terminal Anode
  - Temperature (or flux if known)
  - UDS-0 (electric potential) set to ground voltage
- Terminal Cathode
  - Temperature (or flux if known)
  - UDS-0 (electric potential) is set to the voltage of the cathode (if solving to constant voltage), or the UDS-0 (electric potential) flux is set to the current density in  $A/m^2$ (SI units) (if solving for constant current). Note that the sign of the UDS-0 flux on the cathode side is negative.

## 4.9. Solution Guidelines for the Fuel Cell and Electrolysis Model

For potentiostatic boundary conditions, after initialization, steady-state solutions are calculated easily for cell voltages close to the open-circuit voltage. The same can be said for galvanostatic boundary

conditions and low electric current. By lowering the cell voltage or by raising the average electric current, you can calculate subsequent stationary solutions.

In the event of convergence problems, it is recommended that you change the multigrid cycle to **F-cycle** with **BCGSTAB** (bi-conjugate gradient stabilized method) selected as the stabilization method for the species and the two potential equations. For the species and the user-defined scalar equations, it may be necessary to reduce the termination (criteria) of the multigrid-cycles to  $1\times10^{-3}$ . For stack simulations, the termination criterion may be reduced to  $\times10^{-7}$  for the two potential equations.

Also, it may be useful to turn off **Joule Heating** and **Reaction Heating** in the **Fuel Cell and Electrolysis Models** dialog box (in the **Model** tab) for the first few (approximately 5-10) iterations after initialization. This allows the two electric potentials to adjust from their initial values to more physical values, avoiding the possibility of extreme electrochemical reactions and electric currents that would in turn adversely impact the solution.

## 4.10. Postprocessing the Fuel Cell and Electrolysis Model

You can perform postprocessing using standard ANSYS Fluent quantities and by using user-defined scalars and user-defined memory allocations. By default, the ANSYS Fluent Fuel Cell and Electrolysis Model defines several user-defined scalars and user-defined memory allocations, described in

[Table 4.1: User-Defined Scalar Allocations \(p. 316\)](#) and [Table 4.2: User-Defined Memory Allocations \(p. 316\)](#).

**Table 4.1: User-Defined Scalar Allocations**

UDS 0	Electric Potential (solid phase potential) (Volts)
UDS 1	Protonic Potential (membrane phase potential) (Volts)
UDS 2	Water Saturation (liquid saturation)
UDS 3	Water Content

**Table 4.2: User-Defined Memory Allocations**

UDM 0	X Current Flux Density ( $A/m^2$ )
UDM 1	Y Current Flux Density ( $A/m^2$ )
UDM 2	Z Current Flux Density ( $A/m^2$ )
UDM 3	Current Flux Density Magnitude ( $A/m^2$ )
UDM 4	Ohmic Heat Source ( $W/m^3$ )
UDM 5	Reaction Heat Source ( $W/m^3$ )
UDM 6	Overpotential (Volts)
UDM 7	Phase Change Source ( $kg/m^3-s$ )
UDM 8	Osmotic Drag Coefficient
UDM 9	Liquid Water Activity
UDM 10	Membrane Water Content
UDM 11	Protonic Conductivity (1/ohm-m)
UDM 12	Back Diffusion Mass Source ( $kg/m^3-s$ )

UDM 13	Transfer Current ( $A/m^3$ )
UDM 14	Osmotic Drag Source ( $kg/m^3\cdot s$ )

You can obtain this list by opening the **Execute On Demand** dialog box and pulling down the **Function** drop-down list.



**User-Defined → User-Defined → Execute on Demand...**

and access the execute-on-demand function called `list_pemfc_udf`.

Alternatively, you can view the listing that appears when you first load your Fuel Cell and Electrolysis case, or you can type `list_pemfc_udf` in the text user interface and the listing will appear in the console window.

### Note

- For field variables that are stored in UDM, use the corresponding variables for postprocessing. Postprocessing the UDM itself is not recommended.
- For reviewing simulation results in CFD Post for cases involving UDM or UDS, export the solution data as CFD-Post-compatible file (.cdat) in Fluent and then load this file into CFD-Post.

### Important

When you load older Fuel Cell and Electrolysis cases into ANSYS Fluent, and you are monitoring a UDS using volume or surface monitors, make sure you re-visit the corresponding monitors dialog box (for example, the **Volume Monitor** or the **Surface Monitor** dialog box) to verify that the correct UDS name is used for the appropriate monitor.

## 4.11. User-Accessible Functions

As noted in [Properties \(p. 288\)](#), you can directly incorporate your own formulations and data for the properties of the fuel cell membrane using the `pem_user.c` source code file.

The following listing represents a description of the contents of the `pem_user.c` source code file:

`real Get_P_sat(real T, cell_t c, Thread *t)`

Returns the value of the water vapor saturation pressure as a function of temperature ([Equation 3.41 \(p. 289\)](#)) or other user-specified values.

`real Water_Activity(realP, realT, cell_t c, Thread *t)`

Returns the value of water activity ([Equation 3.39 \(p. 289\)](#)).

`real Water_Content(real act)`

Returns the value of the membrane water content at the membrane catalyst interface ([Equation 3.38 \(p. 289\)](#)).

`real Osmotic_Drag_Coefficient(real P, real T, cell_t c, Thread *t)`

Returns the value of the osmotic drag coefficient ([Equation 3.35 \(p. 289\)](#)).

```
real Membrane_Conductivity(real lam, cell_t c, Thread *t)
```

Returns the value of the membrane's protonic conductivity ([Equation 3.34 \(p. 289\)](#)).

```
real Electrolyte_Conductivity(cell_t c, Thread *t)
```

Returns the value of the ionic conductivity in the electrolyte ([Equation 3.33 \(p. 289\)](#)). (SOFC and Electrolysis only)

```
real Water_Content_Diffusivity(real lam, real T, real mem_mol_dens-  
ity,cell_tc,Thread*t)
```

Returns the value of the water content diffusivity in the membrane ([Equation 3.37 \(p. 289\)](#)).

```
real Gas_Diffusivity(cell_tc, Thread *t, int j_spe)
```

Returns the value of the gaseous species diffusivities in the channels, gas diffusion layers and catalysts ([Equation 3.30 \(p. 288\)](#)).

```
real MCD_Gas_Diffusivity(cell_t c, Thread *t, int i)
```

Returns the tortuosity-corrected value of the gas species diffusion coefficients computed with the multicomponent diffusion option ([Equation 3.32 \(p. 288\)](#)).

```
real Saturation_Diffusivity(real sat, real cos_theta, real porosity, cell_t  
c, Thread *t)
```

Returns the value of diffusivity of the liquid saturation. It comprises the term  $\rho_l \frac{K_{S3}}{\mu_l} \frac{dp_c}{ds}$  from [Equation 3.26 \(p. 287\)](#).

```
real Anode_AV_Ratio(cell_t c, Thread *t)
```

Returns the value of the specific active surface area ( $\zeta_{an}$  in [Equation 3.9 \(p. 285\)](#)) for the anode catalyst.

```
real Cathode_AV_Ratio(cell_t c, Thread *t)
```

Returns the value of the specific active surface area ( $\zeta_{cat}$  in [Equation 3.10 \(p. 285\)](#)) for the cathode catalyst.

```
real Anode_J_TransCoef(cell_t c, Thread *t)
```

Returns the value of the anode reaction reference current density  $\frac{\zeta_{an} j_{an}^{ref}}{([H_2]_{ref})^{\gamma_{an}}}$  used in [Equation 3.9 \(p. 285\)](#).

```
real Cathode_J_TransCoef(cell_t c, Thread *t)
```

Returns the value of the cathode reaction reference current density  $\frac{\zeta_{cat} j_{cat}^{ref}}{([O_2]_{ref})^{\gamma_{cat}}}$  used in [Equation 3.10 \(p. 285\)](#).

```
real Open_Cell_Voltage(cell_t c, Thread *t)
```

Returns the value of the open-circuit voltage  $V_{oc}$  used in [Equation 3.14 \(p. 286\)](#).

```
real Leakage_Current(cell_t c, Thread *t)
```

Returns the value of the leakage current ( $I_{leak}$  in [Equation 3.42 \(p. 290\)](#) through [Equation 3.47 \(p. 290\)](#)).

```
void Set_UDS_Names(char uds[n_uds_required][STRING_SIZE])
```

Used to rename user defined scalars (UDSs). Note that the units of the user defined scalars cannot be changed.

```
void Set_UDS_Names(char uds[n_uds_required][STRING_SIZE])  
{  
    strncpy(uds[0], "Electric Potential", STRING_SIZE-1);  
    strncpy(uds[1], "Protonic Potential", STRING_SIZE-1);  
    strncpy(uds[2], "Water Saturation", STRING_SIZE-1);
```

```

    strncpy(uds[3], "Water Content", STRING_SIZE-1);
}

```

If you want to change the names of UDSs, change the second argument of the `strncpy` functions, recompile and link the module as with any modification to `pem_user.c`. Note that `STRING_SIZE` is fixed in `pem.h` and should not be changed.

### Important

When you load older Fuel Cell and Electrolysis cases into ANSYS Fluent, and you are monitoring a UDS using volume or surface monitors, make sure you re-visit the corresponding monitors dialog box (for example, the **Volume Monitor** or the **Surface Monitor** dialog box) to make sure that the correct UDS name is used for the appropriate monitor.

**void Set\_UDM\_Names(char udm[n\_udm\_required][STRING\_SIZE])**

Used to rename user defined memory (UDMs). Note that the units of user defined memory cannot be changed.

```

void Set_UDM_Names(char udm[n_udm_required][STRING_SIZE])
{
    strncpy(udm[0], "X Current Flux Density", STRING_SIZE-1);
    strncpy(udm[1], "Y Current Flux Density", STRING_SIZE-1);
    strncpy(udm[2], "Z Current Flux Density", STRING_SIZE-1);
    strncpy(udm[3], "Current Flux Density Magnitude", STRING_SIZE-1);
    strncpy(udm[4], "Ohmic Heat Source", STRING_SIZE-1);
    strncpy(udm[5], "Reaction Heat Source", STRING_SIZE-1);
    strncpy(udm[6], "Overpotential", STRING_SIZE-1);
    strncpy(udm[7], "Phase Change Source (PEM)", STRING_SIZE-1);
    strncpy(udm[8], "Osmotic Drag Coefficient (PEM)", STRING_SIZE-1);
    strncpy(udm[9], "Liquid Water Activity (PEM)", STRING_SIZE-1);
    strncpy(udm[10], "Membrane Water Content (PEM)", STRING_SIZE-1);
    strncpy(udm[11], "Protonic Conductivity", STRING_SIZE-1);
    strncpy(udm[12], "Back Diffusion Source (PEM)", STRING_SIZE-1);
    strncpy(udm[13], "Transfer Current", STRING_SIZE-1);
    strncpy(udm[14], "Osmotic Drag Source (PEM)", STRING_SIZE-1);
}

```

If you want to change the names of UDMs, change the second argument of the `strncpy` functions, recompile and link the module as with any modification to `pem_user.c`. Note that `STRING_SIZE` is fixed in `pem.h` and should not be changed.

### Important

When you load older Fuel Cell and Electrolysis cases into ANSYS Fluent, and you are monitoring a UDM using volume or surface monitors, make sure you re-visit the corresponding monitors dialog box (for example, the **Volume Monitor** or the **Surface Monitor** dialog box) to make sure that the correct UDM name is used for the appropriate monitor.

**real electric\_contact\_resistance(face\_t f, Thread \*t, int ns)**

Returns the value for the electrical contact resistance.

**real Transfer\_Current(real i\_ref, real gamma, int species\_i, real alpha\_a, real alpha\_c, real \*dRade, real \*dRcde, Thread \*t, cell\_t c)**

Computes the transfer current ( $A/m^3$ ), corresponding to  $R_{an}$  in [Equation 3.9 \(p. 285\)](#) and  $R_{cat}$  in [Equation 3.10 \(p. 285\)](#).

Inputs for this function include:

i_ref	effective transfer current coefficient, computed by Cathode_J_TransCoef(c,t) or Anode_J_TransCoef(c,t)
gamma	cathode or anode concentration exponent
species_i	species index used in fuel cells (for example i_o2, i_h2, i_h2o)
alpha_a	product of anode exchange coefficient and $\frac{F}{RT}$
alpha_c	product of cathode exchange coefficient and $\frac{F}{RT}$
t	current thread
c	current cell

Outputs for this function include:

source	anode or cathode volumetric transfer current ( $R_{an}$ in <a href="#">Equation 3.9 (p. 285)</a> or $R_{cat}$ in <a href="#">Equation 3.10 (p. 285)</a> )
*dRade	partial derivative of $R_{an}$ with respect to activation loss
*dRcde	partial derivative of $R_{cat}$ with respect to activation loss

**Thermal\_ctk\_pemfc(face\_t f, Thread \*t)**

Used to change the constant value of **Thermal Contact Resistance** for the porous zone set in the **Porous Jump** dialog box to a locally variable value.

For more information, see the following sections:

[4.11.1. Compiling the Customized Fuel Cell and Electrolysis Source Code](#)

This section includes instructions on how to compile a customized Fuel Cell and Electrolysis user-defined module. Note that you can also refer to the file **INSTRUCTIONS-CLIENT** that comes with your distribution (see **addons/fuelcells**).

---

**Important**

It is assumed that you have a basic familiarity with compiling user-defined functions (UDFs). For an introduction on how to compile UDFs, refer to the [Fluent Customization Manual](#).

---

You will first want to use a local copy of the **fuelcells** directory in the **addons** directory before you recompile the Fuel Cell and Electrolysis module.

**4.11.1.1. Compiling the Customized Source Code Under Linux**

1. Make a local copy of the **fuelcells** directory. Do not create a symbolic link.
- 

**Important**

The custom version of the library must be named according to the convention used by ANSYS Fluent: for example, **fuelcells**.

---

2. Change directories to the **fuelcells/src** directory.

3. Make changes to the `pem_user.c` file.
4. Edit the `makefile` located in the `src/` directory and make sure that the `FLUENT_INC` variable correctly refers to the current ANSYS Fluent installation directory. Be careful not to leave any trailing spaces when you make your changes.
5. Define the `FLUENT_ADDONS` environment variable to correspond to your customized version of the Fuel Cell and Electrolysis module.
6. Change directories to the `fuelcells/` directory.
7. Issue the following `make` command:

```
make FLUENT_INC=[ansys_inc/v170/fluent] FLUENT_ARCH=[arch]
-f Makefile-client
```

where `your_arch` is `lnx86` on `LINUX`, or `ultra` on the `Sun` operating system, and so on.

The following example demonstrates the steps required to set up and run a customized version of the Fuel Cell and Electrolysis module that is located in a folder call `home/sample`:

- Make a directory (for example, `mkdir -p /home/sample`).
- Copy the default addon library to this location.

```
cp -RH [ansys_inc/v170/fluent]/fluent17.0.0/addons/fuelcells
/home/sample/fuelcells
```

- Using a text editor, make the appropriate changes to the `pem_user.c` file located in `/home/sample/fuelcells/src/pem_user.c`
- Edit the `makefile` located in the `src/` directory and make sure that the `FLUENT_INC` variable correctly refers to the current ANSYS Fluent installation directory. Be careful not to leave any trailing spaces when you make your changes.
- Build the library.

```
cd /home/sample/fuelcells
make FLUENT_INC=[ansys_inc/v170/fluent] FLUENT_ARCH=[arch]
-f Makefile-client
```

- Set the `FLUENT_ADDONS` environment variable (using CSH, other shells will differ).

```
setenv FLUENT_ADDONS /home/sample
```

- Start ANSYS Fluent and load the customized module using the text interface command.

#### **4.11.1.2. Compiling the Customized Source Code Under Windows**

1. Open **Visual Studio.NET** at the DOS prompt.
2. Make sure that the `$FLUENT_INC` environment variable is correctly set to the current ANSYS Fluent installation directory (for example, `ANSYS Inc\`v170\fluent`).
3. Make a local copy of the `fuelcells` folder. Do not create a shortcut.
4. Enter the `fuelcells\src` folder.

5. Make changes to the `pem_user.c` file.
6. Define the `FLUENT_ADDONS` environment variable to correspond to your customized version of the Fuel Cell and Electrolysis module.
7. Return to the `fuelcells` folder.
8. Issue the following command in the command window:

```
nmake /f makefile_master-client.nt
```

## 4.12. Using the Fuel Cell and Electrolysis Text User Interface

All of the features for the Fuel Cell and Electrolysis Model (sometimes referred to as the Resolved Electrolyte model) that are available through the graphical user interface are also available through text user interface (TUI) commands. The TUI allows text commands to be typed directly in the ANSYS Fluent console window where additional information can be extracted and processed for more advanced analysis.

Once the fuel cell module is loaded (see [Loading the Fuel Cell and Electrolysis Module \(p. 292\)](#)), you can access the text user interface through the console window under `resolved-MEA-fc`. A listing of the various text commands is as follows:

```
resolved-MEA-fuelcells/
  Fuel cell model menu

enable-fc-model?
  Enable/disable fuel cell model

select-model
  Select model

model-options
  Model options

model-parameters
  Model parameters

anode-setup/
  Anode setup

list-zones-briefly
  List zone names and IDs

current-collector
  Set current collector

flow-channel
  Set flow channel

porous-electrode
  Set porous electrode

catalyst-layer
  Set catalyst layer
```

```
cathode-setup/
Cathode setup

list-zones-briefly
List zone names and IDs

current-collector
Set current collector

flow-channel
Set flow channel

porous-electrode
Set porous electrode

catalyst-layer
Set catalyst layer

electrolyte-setup/
Electrolyte setup

electrolyte-layer
Set electrolyte layer

list-zones-briefly
List zone names and IDs

advanced-setup/
Advanced setup

list-zones-briefly
List zone names and IDs

contact-resistivity
Set contact resistivity

coolant-channel
Set coolant channel

stack-management/
Stack setup

list-fc-units
List fuel cell units

list-zones-briefly
List zone names and IDs

create-fc-unit
Create fuel cell unit

modify-fc-unit
Modify fuel cell unit
```

**delete-fc-unit**

Delete fuel cell unit

**set-stack-current-density**

Set the current density on the anode or cathode and modify the current solution to assist convergence. Note: Input here is in units of A/cm<sup>2</sup>. This is only available if the case contains valid data (for example, after initialization, iterating, or reading in data). For more information, see [IV-Curve Calculations Using the Text Interface \(p. 324\)](#).

**set-stack-voltage**

Set the voltage difference in Volts between the anode and the cathode and modify the current solution to assist convergence. This is only available if the case contains valid data (for example, after initialization, iterating, or reading in data). For more information, see [IV-Curve Calculations Using the Text Interface \(p. 324\)](#).

**reset-setup**

Reset the stack setup in case mistakes are made

**submit-setup**

Submit the stack setup and makes the stack setup take effect

**suggest-setup**

Suggest the stack setup, invoking the automatic stack setup

**controls**

Set model control parameters

**reports**

Set electrolyte project area and external contacts

**set-default**

Set default

## 4.12.1. IV-Curve Calculations Using the Text Interface

For valid case and data files, there are two text commands available to assist in the IV-curve calculation. These commands are **set-stack-voltage** (aliased as **ssv**) and **set-stack-current-density** (aliased as **ssc**), available from the Fuel Cell and Electrolysis text command menu: /define/models/resolved-MEA-fc/advanced-setup/stack-management/.

For fuel cells, you either prescribe the voltage and obtain the total current delivered by the fuel cell as a result, or you specify the total current (via flux boundary conditions multiplied by the area) and obtain the voltage as part of the solution. The details of this IV-relation are specific for each single fuel cell and depend on mass and heat transport, electrochemistry and inlet conditions, outlet conditions, operating conditions, and any other parameter or material property involved in the calculation. The IV-curve is important for applications, because its product is the power delivered by the system.

As described earlier in this manual, you would start a new simulation from fairly static conditions, that is, high voltage/low current (which implies low species transport and low heat generation). After convergence, you typically may be interested in solutions for new electric boundary conditions, that is, either for a new cell/stack voltage or current.

In such cases, simply going to the **Boundary Conditions** task page and changing the value of the electric potential (**uds-0**) boundary condition, typically allows only small changes, most notably for

stacks. Otherwise the solution will not converge. This is where the set-stack-voltage and set-stack-current-density commands are important.

In addition to changing the boundary conditions (either to a prescribed voltage or current density), these commands process the current data in order to estimate the solution for the new boundary conditions. Because these commands modify the data, you are prompted to save your data, if you have not already done so.

Before going into details of the commands, here are some general remarks about electric potential boundary conditions.

For fixed voltage boundary conditions, both external contacts have a fixed value for the electric potential ( $uds=0$ ). The anode value will typically be zero, but it does not have to be. The cathode value will be larger than the anode value and the difference ( $V_{cathode} - V_{anode}$ ) is the positive cell/stack voltage.

For a fixed current boundary condition, one external contact has to have a fixed value and the other flux boundary conditions. As described earlier in the manual, typically, the anode will have a fixed (zero) value, and the cathode will be floating, however, you can also set the cathode to a fixed zero potential, yielding a floating negative anode potential.

The set-stack-voltage command sets the effective stack voltage, that is, the difference ( $V_{cathode} - V_{anode}$ ). For fixed voltage boundary conditions for the previous solution, boundary conditions on both boundaries are of type fixed value and then the cathode value will be changed accordingly. In the case of fixed current boundary conditions for the previous solution, the flux boundary condition will be changed to a fixed value boundary condition, and the value adjusted accordingly with respect to the other fixed value boundary condition.

The set-stack-current-density command sets the current density on one boundary to the desired value. Note that the input will be in  $\frac{A}{cm^2}$ , not  $\frac{A}{m^2}$  as you would normally have to enter in the **Boundary Conditions** task page. The reason for this is that average current densities reported in the text command interface are also in  $\frac{A}{cm^2}$ , and this makes it easier to choose the conditions you would like to prescribe next. Also, flux boundary conditions entered in the **Boundary Conditions** dialog box would have to have a positive sign on the anode side, and a negative sign on the cathode side. The input for the text interface command is just a positive number, signs are automatically accounted for.

For fixed current boundary conditions for the previous solution, the set-stack-current-density command changes the respective flux boundary condition accordingly. In the case of fixed voltage boundary conditions for the previous solution, the cathode side is chosen to be changed from a fixed value to a flux boundary condition with the new flux.

The two commands may be mixed in an IV-curve calculation. For the type of boundary condition setups currently described in this manual, boundary condition changes will consistently happen on the cathode side. However, if anode flux boundary conditions had been chosen initially, switching to fixed voltage boundary conditions by set-stack-voltage command and then back to fixed current boundary conditions by the set-stack-current-density command will then have flux boundary conditions on the cathode side. In this case, using the set-stack-current-density command exclusively will preserve the anode flux boundary condition setting.



# Chapter 5: SOFC Fuel Cell With Unresolved Electrolyte Model Theory

This chapter presents an overview of theory and equations for solid oxide fuel cell (SOFC, with unresolved electrolyte) modeling capabilities in ANSYS Fluent.

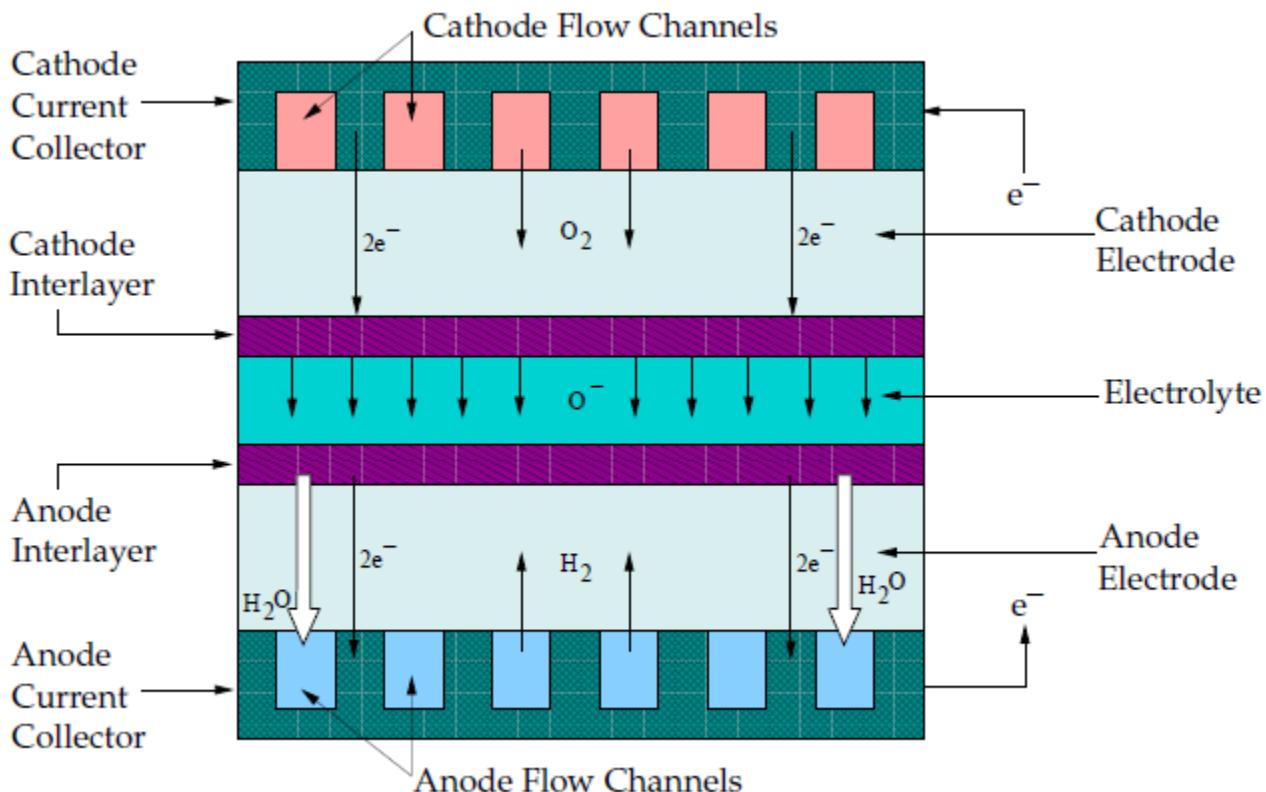
- 5.1. Introduction
- 5.2. The SOFC With Unresolved Electrolyte Modeling Strategy
- 5.3. Modeling Fluid Flow, Heat Transfer, and Mass Transfer
- 5.4. Modeling Current Transport and the Potential Field
- 5.5. Modeling Reactions

## 5.1. Introduction

The Solid Oxide Fuel Cell (SOFC) With Unresolved Electrolyte Model is provided as an add-on module with the standard ANSYS Fluent licensed software.

A fuel cell is an energy conversion device that converts the chemical energy of fuel into the electrical energy. A schematic of a solid oxide fuel cell (SOFC) is shown in [Figure 5.1: Schematic of a Solid Oxide Fuel Cell \(p. 327\)](#).

**Figure 5.1: Schematic of a Solid Oxide Fuel Cell**



As noted in [1] (p. 355), a solid oxide fuel cell is typically composed of an anode, cathode, and an electrolyte. Multiple fuel cells can be connected together, or stacked, using electrical interconnects. The electrolyte material must be solid, that is, non-porous, and exhibit a high ionic conductivity.

Note that the reason this modeling approach is referred to as the "SOFC Model with Unresolved Electrolyte" model is that the anode and the cathode "interlayers" and "electrolyte" (as shown in [Figure 5.1: Schematic of a Solid Oxide Fuel Cell \(p. 327\)](#)) are not actually included in the computational domain. They are modeled as a pair of wall and wall-shadow faces, named "electrolyte interfaces," with the species and energy sources and sinks due to the electrochemical reactions added to the adjacent computational cells.

All components of the fuel cell must have similar thermal expansion in order to minimize thermal stresses, which may cause cracking and de-lamination during thermal cycling. In addition, the components must be chemically stable in order to limit chemical interactions with other cell components.

A solid oxide fuel cell works by having electrically conducting porous ceramic electrodes attached on each side of an ionically conducting ceramic material. At the cathode/electrolyte/gas interface, also known as the triple phase boundary, oxygen is reduced to oxygen ions. The oxygen ions are conducted through the oxygen vacancies in the electrolyte to the anode side. At the anode/electrolyte/gas interface, oxygen ions combine to react with hydrogen at the anode electrode to form water and release electrons. The electrons travel through an external circuit to a load and back to the cathode electrode to close the circuit.

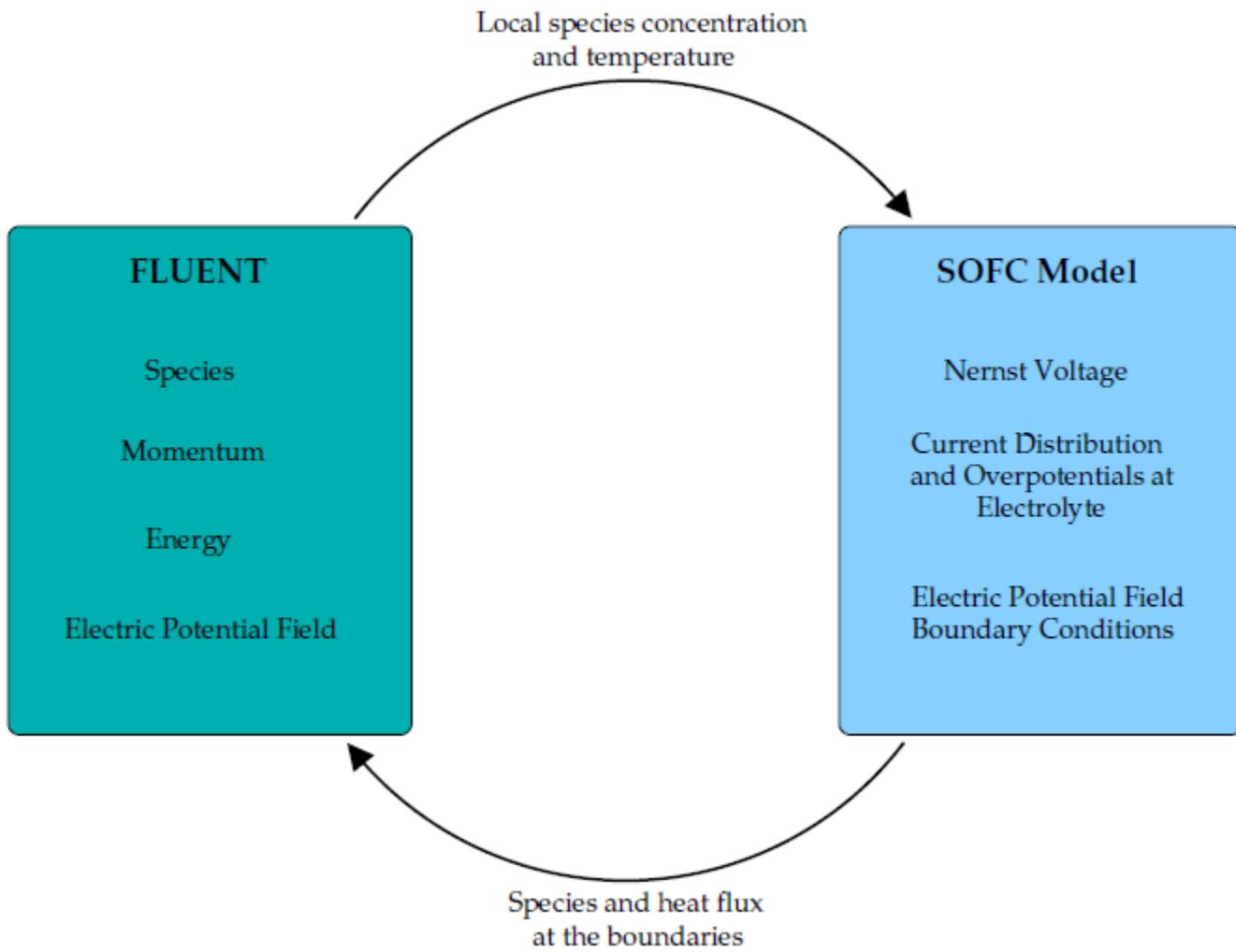
The ANSYS Fluent SOFC With Unresolved Electrolyte Model provides the following features:

- Local electrochemical reactions coupling the electric field and the mass, species, and energy transport.
- Electric field solution in all porous and solid cell components, including ohmic heating in the bulk material.
- Ability to handle  $H_2$  and combined  $CO/H_2$  electrochemistry.
- Inclusion of tortuosity for porous regions
- Treatment of an arbitrary number of electrochemical cells arranged as a stack.
- Significant geometric flexibility for treating planar, tubular, and other nonstandard SOFC configuration.
- Use of non-conformal interface meshing (as long as these interfaces are not the electrolyte interfaces).

## 5.2. The SOFC With Unresolved Electrolyte Modeling Strategy

To model solid oxide fuel cells (SOFC) with unresolved electrolyte, you need to perform the following:

- Capture the fluid flow, heat transfer, and the mass transfer in the flow channels and in the porous anode and cathode electrodes.
- Model the transport of the current and the potential field in the porous electrodes and in the solid conducting regions.
- Model the electrochemical reactions that take place at the electrolyte/electrode/gaseous species interface.

**Figure 5.2: How the SOFC With Unresolved Electrolyte Model Works in ANSYS Fluent**

### 5.3. Modeling Fluid Flow, Heat Transfer, and Mass Transfer

All aspects of fluid flow, heat transfer, and mass transfer in the flow channels and porous electrodes are handled by ANSYS Fluent.

The default multicomponent diffusion model in ANSYS Fluent is used to calculate the mass diffusion coefficient of species  $i$  in the mixture.

To account for the effect of porosity on the multicomponent mass diffusion coefficient

$$D_{ij,eff} = \frac{\varepsilon}{\tau} D_{ij} \quad (5.1)$$

where  $\varepsilon$  is the porosity and  $\tau$  is the tortuosity (that is, the average path length over the actual length).

### 5.4. Modeling Current Transport and the Potential Field

Solving for three-dimensional electrical conduction is directly analogous to the calculation of heat transfer. The potential field throughout the conductive regions is calculated based on the conservation of charge.

$$\nabla \cdot i = 0 \quad (5.2)$$

where

$$i = -\sigma \nabla \phi \quad (5.3)$$

and  $\sigma$  is the electrical conductivity and  $\phi$  is the electrical potential. Therefore, the governing equation for the electric field is the Laplace equation:

$$\nabla \cdot (\sigma \nabla \phi) = 0 \quad (5.4)$$

The electric field potential calculation combines the following attributes:

- Ohmic losses in all the conducting materials, including the electrolyte, electrodes, and current collectors.
- Contact resistance at the appropriate interfaces.
- Ohmic heating through conduction materials as the result of ohmic losses, and the current density throughout the domain.

For more information, see the following sections:

#### 5.4.1. Cell Potential

#### 5.4.2. Activation Overpotential

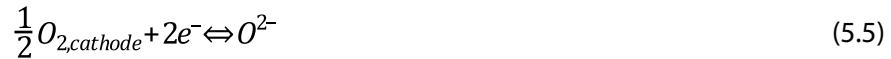
#### 5.4.3. Treatment of the Energy Equation at the Electrolyte Interface

#### 5.4.4. Treatment of the Energy Equation in the Conducting Regions

## 5.4.1. Cell Potential

The electrode reactions are assumed to take place in a single step. The charge transfer reaction acts as the rate limiting step for the electrode reactions.

Oxygen is electrochemically reduced at the triple phase boundary at the cathode electrode:



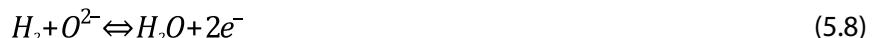
Oxygen is electrochemically re-oxidized at the triple phase boundary at the anode electrode:



In the absence of an electrical load, the oxygen activity on both sides of the electrolyte is fixed and given by their respective chemical potentials. Under equilibrium, the electromotive force, or reversible cell voltage, is given by the Nernst equation:

$$\phi_{ideal} = \frac{RT}{4F} \ln \frac{p_{O_{2,cathode}}}{p_{O_{2,anode}}} \quad (5.7)$$

If hydrogen is present at the anode electrode, then the cell reaction becomes:



At equilibrium, the cell voltage is given by the Nernst equation:

$$\phi_{ideal} = \phi^o + \frac{RT}{2F} \ln \frac{p_{H_2} p_{O_2}^{\frac{1}{2}}}{p_{H_2O}} \quad (5.9)$$

where  $\phi^o$  is the open-circuit cell voltage.

The cell potential measured at equilibrium (that is, no load), is called the open circuit voltage. The open circuit voltage should be equivalent to the Nernst potential at no load, unless there is leakage across

the electrolyte. When the external circuit is closed, then cell voltage drops due to polarization losses at the electrodes.

The electric field and the electrochemistry interact solely at the electrolyte interface. ANSYS Fluent treats the electrolyte interface as an impermeable wall. The potential field must have a "jump" condition applied to the two sides of this wall to account for the effect of the electrochemistry. To closely couple the electrochemical behavior to the potential field calculation, you need to include all of the electrochemical effects into this jump condition. It encapsulates the voltage jump due to Nernst, the voltage reduction due to activation, the Ohmic losses due to the resistivity of the electrolyte, and a linearized form for voltage reduction due to activation. This interface condition relates the potential on the anode side and the cathode side of the electrolyte and has the following form:

$$\phi_{cell} = \phi_{jump} - \eta_s \quad (5.10)$$

where

$$\phi_{jump} = \phi_{ideal} - \eta_{ele} - \eta_{act,a} - \eta_{act,c} \quad (5.11)$$

where  $\eta_{ele}$  represents the ohmic overpotential of the electrolyte, and  $\eta_{act,a}, \eta_{act,c}$  represent the activation overpotential of the anode and the cathode.  $\eta_s$  represent ohmic losses in the solid conducting regions.  $\phi_{ideal}$  represents the Nernst potential.

## 5.4.2. Activation Overpotential

The general electrochemical reaction is, according to [7] (p. 355),



where  $a_j$  is the stoichiometric coefficient of species  $j$ ,  $A_j$  is the chemical species, and  $n$  is the number of electrons.

The reaction rate is:

$$r = \frac{i}{nF} = k_a e^{\frac{\alpha_a n F}{RT} \phi} \prod_i c_i^{p_i} - k_c e^{-\frac{\alpha_c n F}{RT} \phi} \prod_i c_i^{q_i} \quad (5.13)$$

where  $\phi$  is the voltage,  $k_a$  and  $p_i$  are the rate constant and the reaction order for the anodic direction,  $k_c$  and  $q_i$  are the rate constant and the reaction order for the cathodic direction,  $\alpha_a$  is the anodic transfer coefficient,  $\alpha_c$  is the cathodic transfer coefficient, and  $n$  is the number of electrons that are released. At equilibrium, the forward and the backward reaction rates are the same, therefore:

$$\frac{i_0}{nF} = k_a e^{\frac{\alpha_a n F}{RT} \phi_0} \prod_i c_i^{p_i} = k_c e^{-\frac{\alpha_c n F}{RT} \phi_0} \prod_i c_i^{q_i} \quad (5.14)$$

where  $i_0$  is the exchange current density.

The reaction rate (that is, current) can be written in terms of the exchange current density  $i_0$  to obtain the Butler-Volmer formulation [7] (p. 355):

$$i = i_0 \left[ e^{\frac{\alpha_a n (\phi - \phi_0) F}{RT}} - e^{-\frac{\alpha_c n (\phi - \phi_0) F}{RT}} \right] \quad (5.15)$$

The activation overpotential is the energy lost due to the slowness of electrochemical reactions at the anode and the cathode electrodes.

$$\eta_{act} = \phi - \phi_0 \quad (5.16)$$

Using this relation, the Butler-Volmer equation can be written as:

$$i = i_{0eff} \left[ e^{\frac{\alpha_a m_{act} F}{RT}} - e^{-\frac{\alpha_c m_{act} F}{RT}} \right] \quad (5.17)$$

where

$$i_{0eff} = i_{0,ref} \left( \frac{\chi_j}{\chi_{j,ref}} \right)^{\gamma_j} \quad (5.18)$$

with  $i_{0,ref}$  being the exchange current density at the reference condition,  $(\chi_j)$  is the mole fraction and  $\gamma_j$  is the concentration exponent for species  $j$ . More specifically, at the anode side, you have:

$$i_{0eff}^{anode} = i_{0,ref}^{anode} \left( \frac{\chi_{H_2}}{\chi_{H_2,ref}} \right)^{\gamma_{H_2}} \left( \frac{\chi_{H_2O}}{\chi_{H_2O,ref}} \right)^{\gamma_{H_2O}} \quad (5.19)$$

Likewise, at the cathode side, you have:

$$i_{0eff}^{cathode} = i_{0,ref}^{cathode} \left( \frac{\chi_{O_2}}{\chi_{O_2,ref}} \right)^{\gamma_{O_2}} \quad (5.20)$$

Given values for  $\alpha_a$  and  $\alpha_c$ , the full version of the Butler-Volmer equation can be solved using the Newton method, therefore finding the activation overpotential at the anode ( $\eta_{act,a}$ ) and the cathode ( $\eta_{act,c}$ ).

### 5.4.3. Treatment of the Energy Equation at the Electrolyte Interface

For an incompressible flow, the energy equation that ANSYS Fluent solves for within each computational cell is given by the following:

$$\frac{\partial}{\partial t} (\rho E) + \nabla \cdot (\vec{v}(\rho E + p)) = \nabla \cdot \left( k_{eff} \nabla T - \sum_j h_j \vec{j}_j + (\bar{\tau}_{eff} \cdot \vec{v}) \right) + S_h \quad (5.21)$$

where  $S_h$  is the volumetric source or sink of energy and where

$$E = h - \frac{p}{\rho} + \frac{v^2}{2} \quad (5.22)$$

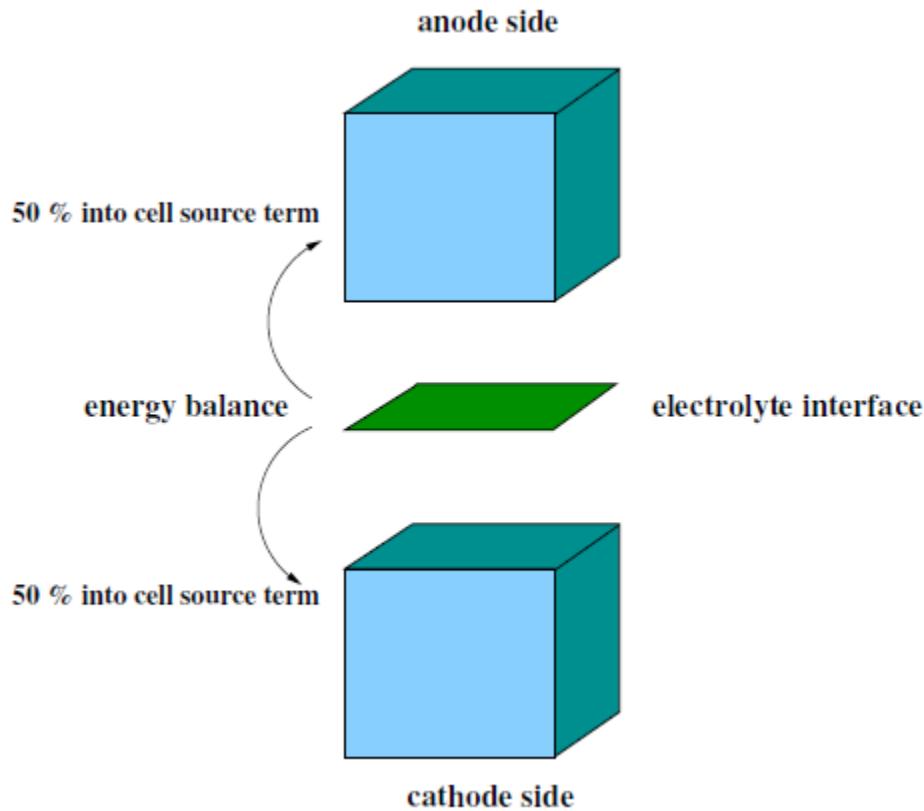
and

$$h = \sum_j Y_j h_j \quad (5.23)$$

In all electrically conducting zones (for example, electrodes, current collectors, interconnects), ohmic heating,  $i^2 * R_{ohmic}$ , is added to the energy equation as a source term. In other words,

$$S_h = i^2 * R_{ohmic} \quad (5.24)$$

In addition, the energy equation needs treatment at the electrode-electrolyte interface to account for the heat generated or lost as the result of electrochemistry and the overpotentials (that is, activation overpotential and ohmic loss through the electrolyte).

**Figure 5.3: Energy Balance at the Electrolyte Interface**

The total energy balance on the electrolyte interface is computed by enumerating the enthalpy flux of all species, including the heat of formation (sources of chemical energy entering the system), and then subtracting off the work done (leaving the system) which is simply the local voltage jump multiplied by the local current density. What remains is the waste heat due to irreversibilities. For hydrogen reaction, the balance would be

$$\dot{Q}'' = \dot{h}_{H_2}'' + \dot{h}_{O_2}'' - \dot{h}_{H_2O}'' - i\Delta V \quad (5.25)$$

where  $\dot{Q}$  is the heat generation (W) and  $\dot{h}$  is the total enthalpy of species (J/s) composed of the sensible enthalpy in addition to the enthalpy of formation.

The heat of formation is

$$h_{H_2} = \dot{m}_{H_2} \left[ \int_{T_{ref}}^T C_p dT + h_0 \right] \quad (5.26)$$

The source term is then added in the cell energy equation by taking  $S_h = \frac{\dot{Q}}{\text{Volume}}$ .

One half of this value is applied as a source term to the energy equation of the anode computational cell adjacent to the electrolyte and the other half is applied as a source term to the energy equation for the cathode cell adjacent to the electrolyte. The equal distribution of the heat generation/destruction is purely arbitrary. Note that by using the work term, the effect from all overpotentials are taken into account.

## 5.4.4. Treatment of the Energy Equation in the Conducting Regions

Ohmic polarization involves ionic losses through the electrolyte, electrical resistance in the conducting porous electrodes and solid collectors. The ohmic polarization also includes the electrical resistance at the interface of the current collectors and the electrodes or the electrodes and the membrane (that is, the contact resistance).

$$\eta_{ohmic} = i \cdot R \quad (5.27)$$

## 5.5. Modeling Reactions

The ANSYS Fluent SOFC With Unresolved Electrolyte Model can model both electrochemical reactions, as well as *CO* electrochemistry. Reforming reactions that generate hydrogen can be modeled by the standard volumetric reaction mechanism in ANSYS Fluent.

For more information, see the following sections:

- 5.5.1. Modeling Electrochemical Reactions
- 5.5.2. Modeling CO Electrochemistry

### 5.5.1. Modeling Electrochemical Reactions

The rate of species production and destruction is:

$$S = -\frac{ai}{nF} \left( g-mole / m^2 / sec \right) \quad (5.28)$$

where  $S$  is the source or sink of the species (molar flux),  $a$  is the stoichiometric coefficient,  $i$  is the current density ( $A / m^2$ ),  $n$  is the number of electrons per mole of fuel, and  $F$  is the Faraday constant.

Using the local current information, the ANSYS Fluent SOFC With Unresolved Electrolyte Model applies species fluxes to the electrode boundaries. By convention [7] (p. 355), the current density is positive when it flows from the electrode into the electrolyte solution. The current densities are positive at the anodes and negative at the cathodes.

The reaction at the cathode electrode is:

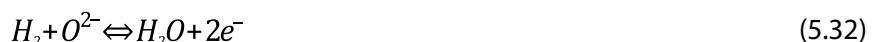


or

$$S_{O_2} = -\frac{-\left(\frac{1}{2}\right)(-i)}{2F} = -\frac{i}{4F} \quad (5.30)$$

$$S_{O^{2-}} = -\frac{(-i)}{2F} = -\frac{i}{2F} \quad (5.31)$$

The reaction at the anode electrode is:



$$S_{H_2} = -\frac{i}{2F} \quad (5.33)$$

$$S_{O^{2-}} = -\frac{i}{2F} \quad (5.34)$$

$$S_{H_2O} = -\frac{-\left(1\right)\left(i\right)}{2F} = \frac{i}{2F} \quad (5.35)$$

Note that the total enthalpy includes the formation enthalpy for each species only when the volumetric reactions are enabled. Once the volumetric reactions are enabled, the SOFC model solver can correctly compute the reaction heat.

## 5.5.2. Modeling CO Electrochemistry

In practice, if carbon monoxide (*CO*) is present in the anode fuel stream, it may be oxidized to generate carbon dioxide (*CO*<sub>2</sub>). This effect is modeled by introducing a *H*<sub>2</sub>/ *CO* split factor as follows:

$$\alpha = \frac{x_{H_2}}{x_{H_2} + x_{CO}} \quad (5.36)$$

where  $x_{H_2}$  and  $x_{CO}$  are species mole fractions of *H*<sub>2</sub> and *CO*, respectively.

The molar source terms for each anode-side species are:

$$S_{H_2} = -\frac{\alpha}{2F} i \quad (5.37)$$

$$S_{CO} = -\frac{1-\alpha}{2F} i \quad (5.38)$$

$$S_{CO_2} = \frac{1-\alpha}{2F} i \quad (5.39)$$

$$S_{H_2O} = \frac{\alpha}{2F} i \quad (5.40)$$

By default, [Equation 5.36 \(p. 335\)](#) is used to compute the *H*<sub>2</sub>/ *CO* split factor, however, you can define your own split factor in the user-defined function called `h2_co_split_func()` (see [User-Accessible Functions for the Solid Oxide Fuel Cell With Unresolved Electrolyte Model \(p. 351\)](#)).



---

## Chapter 6: Using the Solid Oxide Fuel Cell With Unresolved Electrolyte Model

---

The procedure for setting up and solving solid oxide fuel cell (SOFC) problems (with unresolved electrolyte) is described in detail in this chapter. Refer to the following sections for more information:

- [6.1. Limitation on Modeling Solid Oxide Fuel Cells](#)
- [6.2. Installing the Solid Oxide Fuel Cell With Unresolved Electrolyte Model](#)
- [6.3. Loading the Solid Oxide Fuel Cell With Unresolved Electrolyte Module](#)
- [6.4. Solid Oxide Fuel Cell With Unresolved Electrolyte Module Set Up Procedure](#)
- [6.5. Setting the Parameters for the SOFC With Unresolved Electrolyte Model](#)
- [6.6. Setting Up the Electrochemistry Parameters](#)
- [6.7. Setting Up the Electrode-Electrolyte Interfaces](#)
- [6.8. Setting Up the Electric Field Model Parameters](#)
- [6.9. User-Accessible Functions for the Solid Oxide Fuel Cell With Unresolved Electrolyte Model](#)
- [6.10. Using the Solid Oxide Fuel Cell With Unresolved Electrolyte Text User Interface](#)

### 6.1. Limitation on Modeling Solid Oxide Fuel Cells

The anisotropic species diffusivity option is not compatible with the Solid Oxide Fuel Cell model.

### 6.2. Installing the Solid Oxide Fuel Cell With Unresolved Electrolyte Model

The Solid Oxide Fuel Cell (SOFC) With Unresolved Electrolyte Model is provided as an addon module with the standard ANSYS Fluent licensed software. The module is installed with the standard installation of ANSYS Fluent in a directory called `addons/sofc` in your installation area. The SOFC With Unresolved Electrolyte Model consists of a UDF library and a pre-compiled scheme library, that must be loaded and activated before calculations can be performed.

### 6.3. Loading the Solid Oxide Fuel Cell With Unresolved Electrolyte Module

The Solid Oxide Fuel Cell (SOFC) With Unresolved Electrolyte Model is loaded into ANSYS Fluent through the text user interface (TUI). The module can only be loaded after a valid ANSYS Fluent mesh or case file has been set or read. The text command to load the addon module is

`define → models → addon-module`

A list of ANSYS Fluent addon modules is displayed:

```
> /define/models/addon-module
Fluent  Addon Modules:
 0. None
 1. MHD Model
 2. Fiber Model
 3. Fuel Cell and Electrolysis Model
 4. SOFC Model with Unresolved Electrolyte
 5. Population Balance Model
 6. Adjoint Solver
 7. Battery Module
```

```
8. MSMD Battery Model  
9. PEM Fuel Cell Model  
Enter Module Number: [0] 4
```

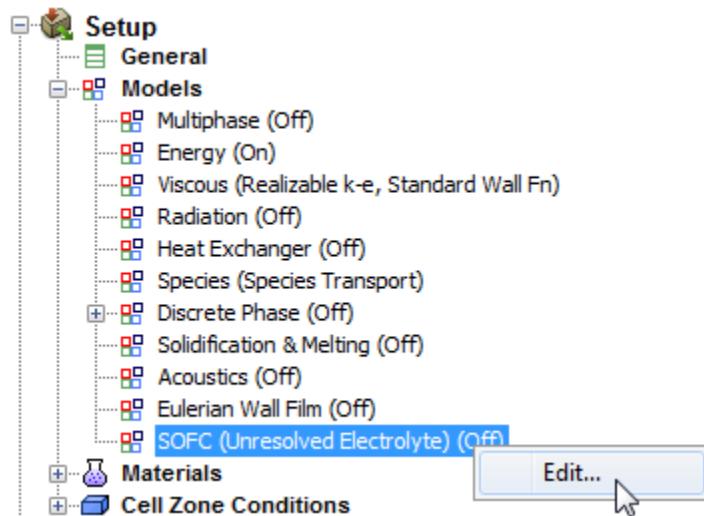
Select the SOFC With Unresolved Electrolyte Model by entering the module number 4. During the loading process, a scheme library (containing the graphical and text user interface) and a UDF library (containing a set of user defined functions) are loaded into ANSYS Fluent.

Once the module has been loaded, the **SOFC Model** **SOFC Model (Unresolved Electrolyte)** option appears in the tree under the **Models** branch and in the **Models** task page.

To enable the SOFC model: In the tree under the **Models** branch, right-click **Fuel Cells and Electrolysis** and select **Edit...** in the menu that opens.



**Figure 6.1: Opening the SOFC Model Dialog Box in the Tree**



When the **SOFC Model** dialog box appears, select the **Enable SOFC Model** option.

## 6.4. Solid Oxide Fuel Cell With Unresolved Electrolyte Module Set Up Procedure

The following describes an overview of the procedure required in order to use the SOFC With Unresolved Electrolyte Model in ANSYS Fluent.

1. Start ANSYS Fluent.

You must start ANSYS Fluent in 3D double-precision mode. Note that the SOFC With Unresolved Electrolyte Model is only available in 3D.

2. Read the case file.



3. Scale the mesh.

**Setup** → **General** → **Scale...**

4. Define various model parameters for the simulation.

a. Check the solver settings.

**Setup** → **General**

- i. In the **Solver** group box, enable **Pressure Based** under **Type**.
- ii. Enable **Steady** under **Time**.
- iii. Enable **Absolute** under **Velocity Formulation**.

b. In the **Energy** dialog box, enable the **Energy** option (if it is not already enabled).

**Setup** → **Models** → **Energy** **ON**

c. In the **Viscous Model** dialog box, enable the **Laminar** option (if it is not already enabled).

**Setup** → **Models** → **Viscous** **Edit...**

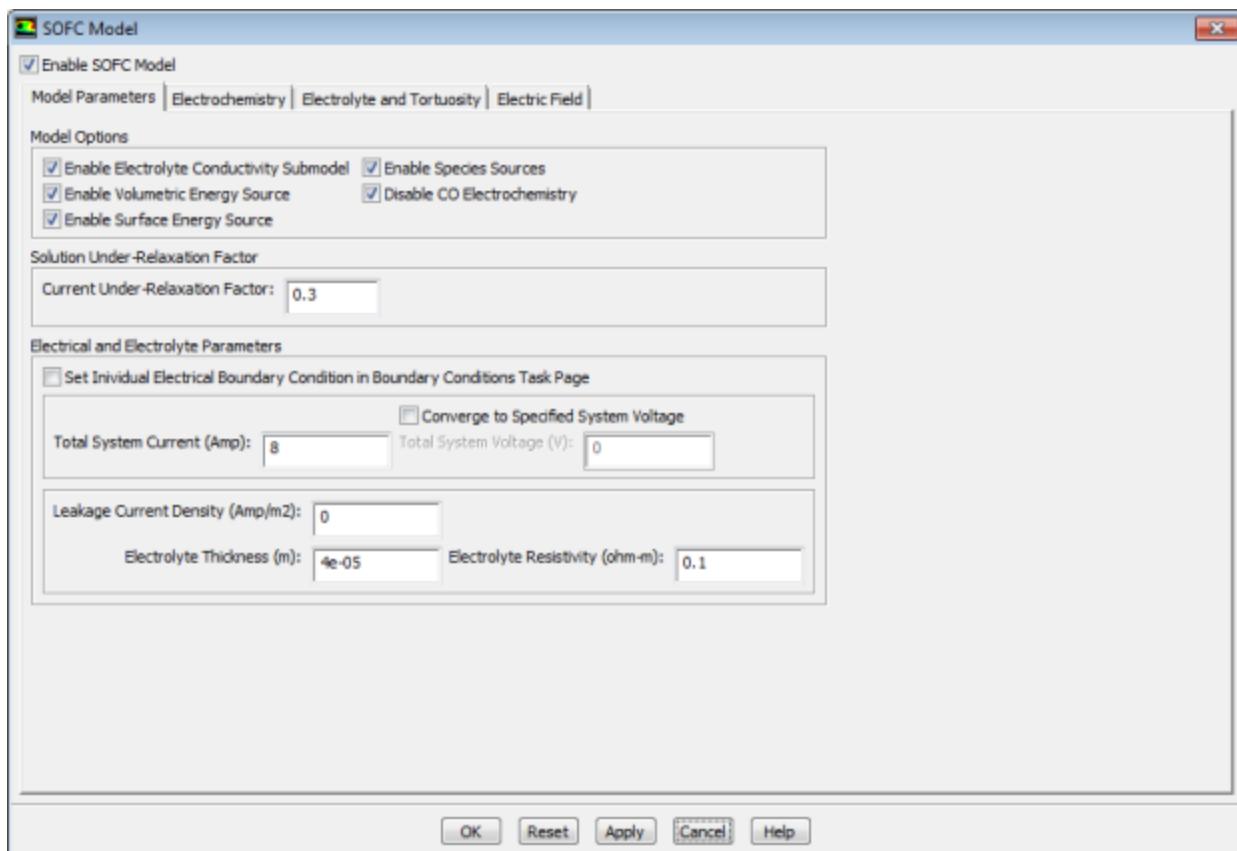
d. In the **Species Model** dialog box, configure the following settings:

**Setup** → **Models** → **Species** **Edit...**

- i. Enable the **Species Transport** option.
- ii. Enable the **Volumetric** option under **Reactions** (see [Enabling Species Transport and Reactions and Choosing the Mixture Material in the \*Fluent User's Guide\*](#) for details).
- iii. From the **Mixture Material** drop-down list, select an appropriate mixture material or create your own mixture material from `mixture-template` as described in [Enabling Species Transport and Reactions and Choosing the Mixture Material in the \*Fluent User's Guide\*](#). If you consider H<sub>2</sub>/CO electrochemistry, ensure that your mixture material include H<sub>2</sub>, CO and CO<sub>2</sub>.
- iv. Disable the **Inlet Diffusion** option under **Options** (see [Diffusion at Inlets in the \*Fluent Theory Guide\*](#)).
- v. Enable the **Diffusion Energy Source** option under **Options** (see [Treatment of Species Transport in the Energy Equation in the \*Fluent Theory Guide\*](#)).
- vi. Enable the **Full Multicomponent Diffusion** option under **Options** (see [Mass Diffusion in Laminar Flows in the \*Fluent Theory Guide\*](#)).
- vii. Enable the **Thermal Diffusion** option under **Options** (see [Mass Diffusion in Laminar Flows in the \*Fluent Theory Guide\*](#)).

e. Enable the **SOFC Model** model and set the parameters for the SOFC with unresolved electrolyte model:

**Setup** → **Models** → **SOFC Model (Unresolved Electrolyte)** **Edit...**



- i. In the **Model Parameters** tab, set the **Current Under-Relaxation Factor** to a value of either 0 . 3 or 0 . 4.
- ii. Under **Model Options**, select the appropriate SOFC model options.
- iii. Set the **Electrical and Electrolyte Parameters** according to your problem specification.
- iv. Select the **Enable Surface Energy Source** option, the **Enable Species Sources** option, and the **Disable CO Electrochemistry** option.

If the electrolyte resistivity changes as a function of temperature, then turn on the **Enable Electrolyte Conductivity Submodel**

**Table 6.1: User-Defined Memory Allocations**

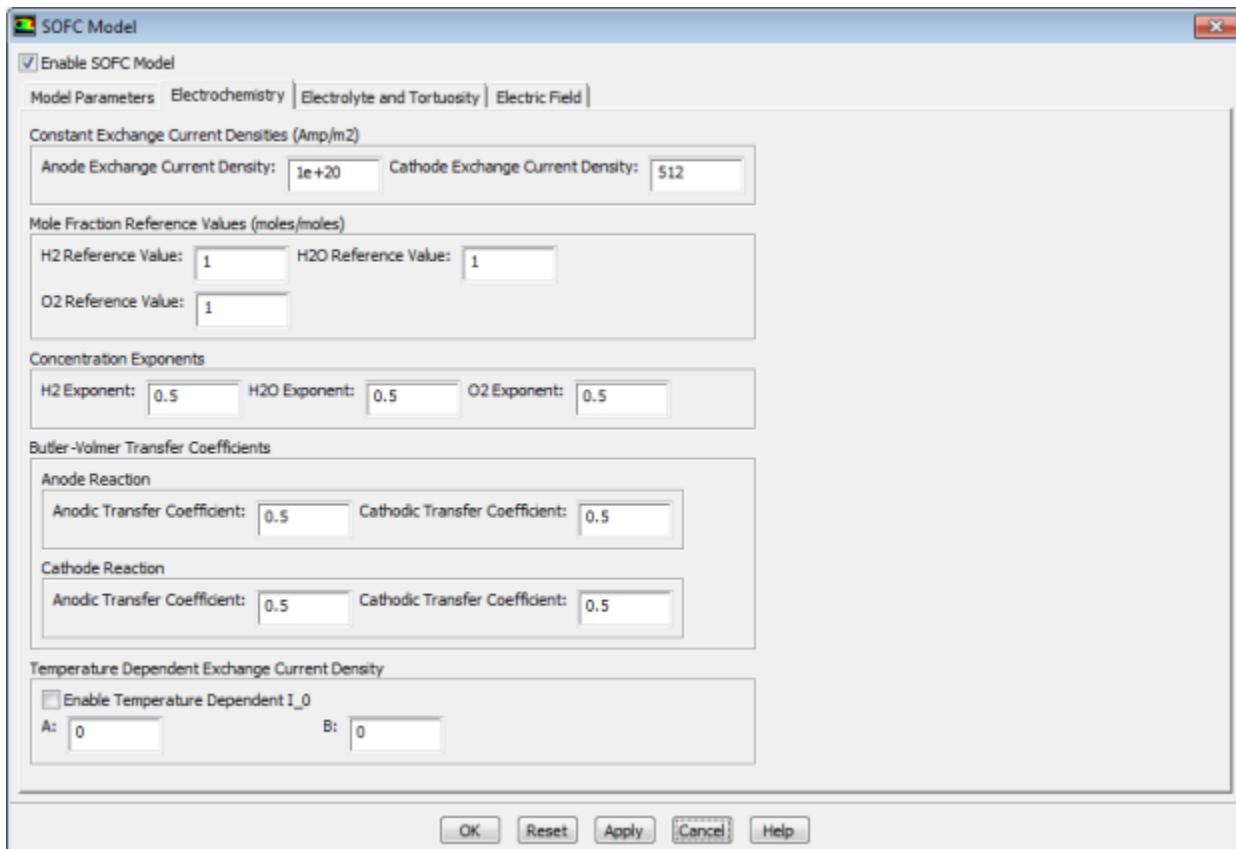
UDM-0	Interface Current Density ( $A/m^2$ )
UDM-1	Nernst Potential (Volts)
UDM-2	Activation Overpotentail (Volts)
UDM-3	Volumetric Ohmic Source ( $W/m^3$ )
UDM-4	x Component of the Current Density ( $A/m^2$ )
UDM-5	y Component of the Current Density ( $A/m^2$ )
UDM-6	z Component of the Current Density ( $A/m^2$ )
UDM-7	Electrolyte Voltage Jump
UDM-8	Electrolyte Resistivity (Ohm-m)

UDM-9	Effective Electric Resistance
UDM-10	Anode Activation
UDM-11	Cathode Activation
UDM-12	Electrochemical Source ( $W/m^3$ )
UDM-13	Magnitude of Current Density ( $A/m^2$ )

Note that UDM-7 and UDM-8 are the linearized values that ANSYS Fluent uses to solve the potential field and electrochemical coupling. They are necessary to the calculations but do not contain any real physical meaning.

Note that UDM-10 and UDM-11 contain the disaggregated activation polarizations at the anode and cathode.

- f. In the **Electrochemistry** tab, set the electrochemistry parameters.

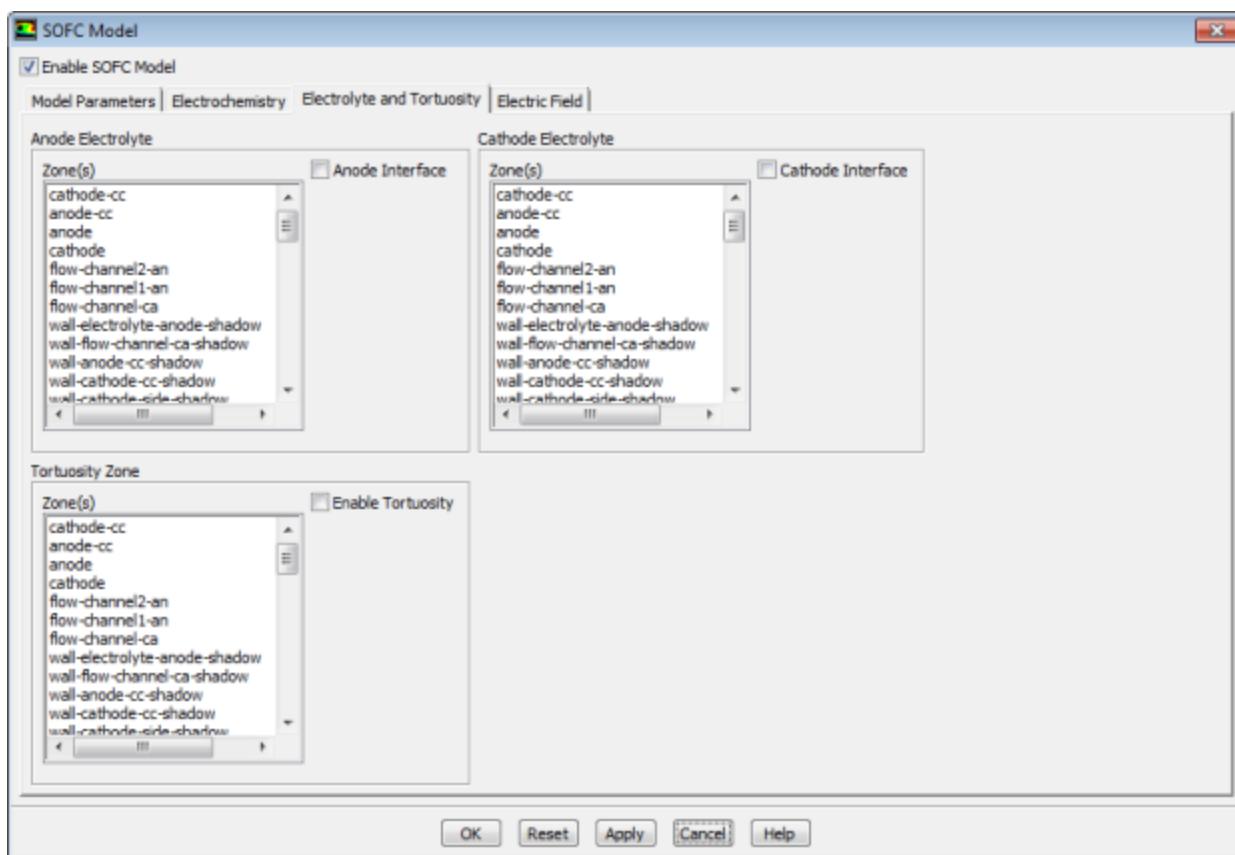


- i. Enter values for the **Constant Exchange Current Densities** for the anode and the cathode. These are the  $i_o$  values in the Butler-Volmer equation (Equation 5.17 (p. 332)) for the anodic and cathodic reactions. By default, the **Anode Exchange Current Density** is set to 1000 Amps and the **Cathode Exchange Current Density** is set to 100 Amps.
- ii. Enter values for the **Mole Fraction Reference Values**.

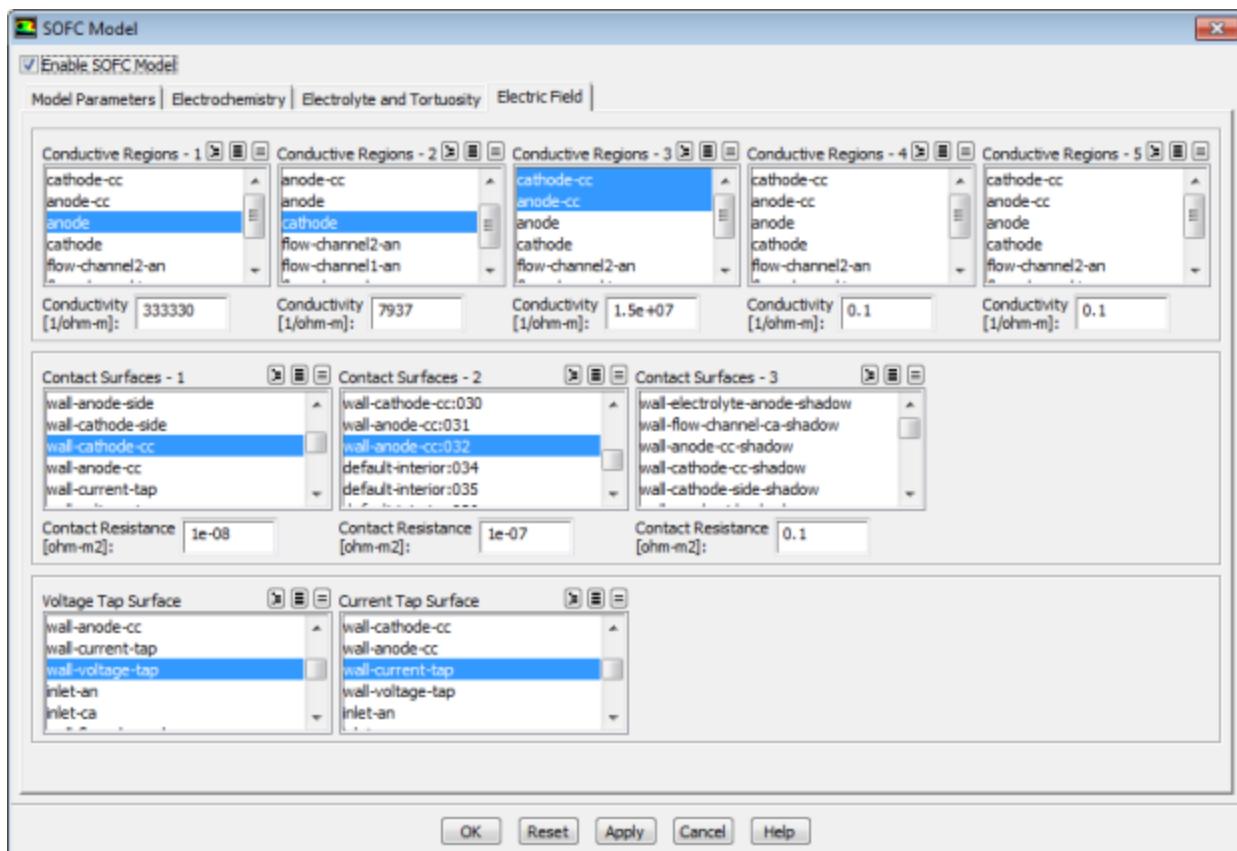
These are the species concentrations at which the exchange current densities were taken (Equation 5.18 (p. 332)-Equation 5.20 (p. 332)). These are used to adjust the  $i_o$  values of reactant species that are depleted. By default, the **H2 Reference Value** is set to 0.8 moles/moles,

the **O2 Reference Value** is set to 0.21 moles/moles, and the **H2O Reference Value** is set to 0.2 moles/moles.

- iii. Enter values for the **Stoichiometric Exponents** by setting values for the **H2 Exponent**, the **H2O Exponent**, and the **O2 Exponent** (defaulted to 0.5). These are the stoichiometric factors in the electrochemical reaction equation. They are used as exponents as part of the  $i_0$  scaling (Equation 5.19 (p. 332) and Equation 5.20 (p. 332)).
  - iv. Enter values for the **Butler-Volmer Transfer Coefficients** by setting values for the **Anodic Transfer Coefficient** ( $\alpha_a$  in Equation 5.17 (p. 332)) and the **Cathode Transfer Coefficient** ( $\alpha_c$  in Equation 5.17 (p. 332)) for both the anode reaction and the cathode reaction (defaulted to 0.5). These are the alpha values in the Butler-Volmer equation (Equation 5.17 (p. 332)). They represent the forward and backward rates of reaction at both the anode and cathode.
  - v. Enter values for the **Temperature Dependent Exchange Current Density** by turning on the **Enable Temperature Dependent  $I_0$**  option and setting values for **A** and **B** (Equation 6.3 (p. 349)). These two coefficients allow the  $i_0$  for the cathode to vary as a function of temperature.
- g. In the **Electrode and Tortuosity** tab, set the anode and cathode interface components and set the tortuosity parameters for the SOFC With Unresolved Electrolyte Model.



- i. Under **Anode Electrolyte**, select a zone in the **Zone(s)** list and enable **Anode Interface** if it is applicable.
  - ii. Do the same for **Cathode Electrolyte** and **Tortuosity Zone**.
- h. In the **Electric Field** tab, set the parameters for the electric field model.



- i. Select up to 5 conductive regions.
  - ii. Select up to 3 contact surfaces.
  - iii. Select a voltage tap surface.
  - iv. Select a current tap surface.
5. Define material properties.
- a. Create or re-define new solid materials as appropriate for the anode, the cathode, and the electrolyte according to your problem specification.

### Important

Note that although the ANSYS Fluent SOFC With Unresolved Electrolyte Model does support the shell conduction model that you use to take into account the transversal conductive heat inside the electrolyte material, it does not currently support a similarly transversal conduction of electric current inside the electrolyte.

- b. Edit the mixture-template mixture material.

**Setup** → **Materials** → **Create/Edit...**

- i. In the **Create/Edit Materials** dialog box, click **Fluent Database...** to open the **Fluent Database Materials** dialog box.

- ii. In the **Fluent Database Materials** dialog box, make a copy of the **h2** fluid material.
- iii. In the **Create/Edit Materials** dialog box, change the **Material Type** to **mixture** and click **Edit...** for the **Mixture Species**.
- iv. In the **Species** dialog box, arrange the materials under **Selected Species** in the following order: **h2o, o2, h2, and n2**. (The species ordering is not important as long as **n2** is the last species.)
- v. In the **Create/Edit Materials** dialog box, change the **Thermal Conductivity** and the **Viscosity** to **ideal-gas-mixing-law**.
- vi. Change the **Mass Diffusivity** to **user-defined** and select **diffusivity::sofc** as the corresponding user-defined function.
- vii. Change the **UDS Diffusivity** to **user-defined** and select **E\_Conductivity::sofc** as the corresponding user-defined function.
- viii. Retain the default values for the other parameters and click **Change/Create**.

6. Set the operating conditions.

 **Setup** →  **Boundary Conditions** → **Operating Conditions...**

Retain the default values.

7. Set the boundary conditions.

 **Setup** →  **Boundary Conditions**

Define the conditions at the anode, the cathode, the interface between the anode and current collector, the interface between the cathode and the current collector, the anode inlet, and the cathode inlet boundaries according to your problem specification.

Note that sources only need to be hooked to the mass, species, and energy equation in a single fluid zone in order for the ANSYS Fluent SOFC With Unresolved Electrolyte Model to function properly (but can be hooked to all zones if you so choose). ANSYS Fluent does not rely on cell or thread referencing in the sources in order to cover the solution domain.

8. Set the multigrid control parameters.

 **Solution** → **Solution Controls**  **Advanced...**

- a. In the **Multigrid** tab of the **Advanced Solution Controls** dialog box, set the cycle type for **h2, h2o, and o2** to **V-cycle**. For serial calculations, set the cycle type for **Energy** and **User-defined Scalar-0** to **W-cycle** or **F-cycle**. For parallel calculations, select the **F-cycle** option for both.
- b. Set the **Max Cycles** to **50**.
- c. Retain the default values for the rest of the parameters.

9. Define the convergence criteria.

 **Solution** → **Monitors** → **Residuals**  **Edit...**

In the **Residual Monitors** dialog box, set the **Convergence Criterion** for all equations to 1e-08.

10. Initialize the flow field.

 **Solution** →  **Solution Initialization** → **Initialize**

Retain the default values for all parameters.

The current density in UDM-0 is a conservative flux of electricity ( $A/m^2$ ). Performing an area integral over the electrolyte surface will sum to the total current ( $A$ ). That value is computed at the electrolyte faces during the electric field solution. The values in UDM-4 through UDM-6 contain cell-centered values of the current density vector. These are not, and cannot be, conservative, so depending on the material conductivity and the geometric configuration, these can sometimes unavoidably produce values that do not match the UDM values. The same issues exist when interpolating velocity values to obtain the mass fluxes.

Note that, by default, the ANSYS Fluent SOFC With Unresolved Electrolyte Model defines a single user-defined scalar:

**Table 6.2: User-Defined Scalar Allocations**

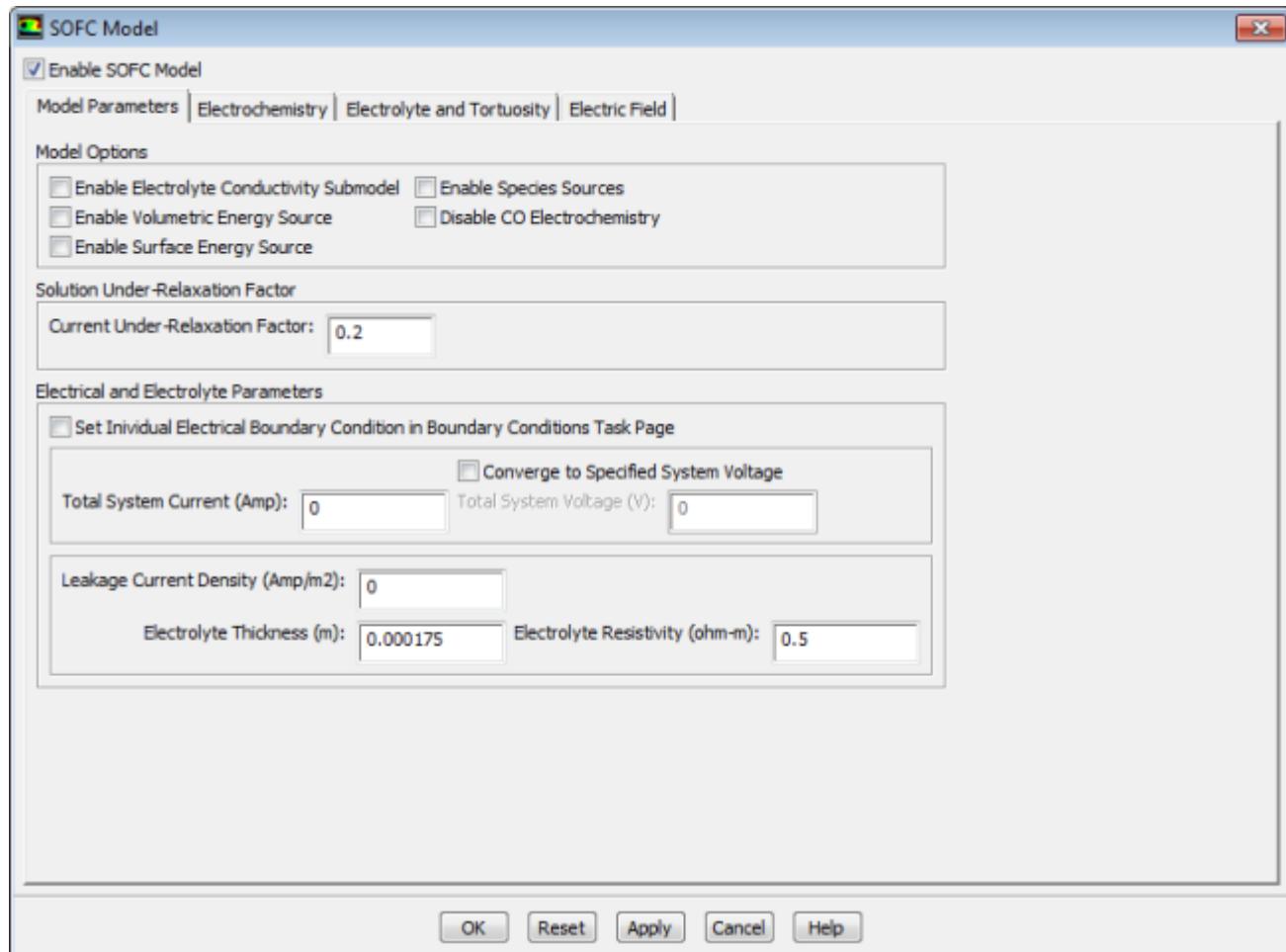
UDS-0	Electric Potential (Volts)
-------	----------------------------

#### Note

- For field variables that are stored in UDM, use the corresponding variables for post processing. Postprocessing the UDM itself is not recommended.
- For reviewing simulation results in CFD Post for cases involving UDM or UDS, export the solution data as CFD-Post-compatible file (. cdat) in Fluent and then load this file into CFD-Post.

## 6.5. Setting the Parameters for the SOFC With Unresolved Electrolyte Model

You can specify the general settings for the SOFC With Unresolved Electrolyte Model using the **Model Parameters** tab in the **SOFC Model** dialog box.

**Figure 6.2: The Model Parameters Tab in the SOFC Model Dialog Box**

From this tab, you can set the various parameters for the SOFC With Unresolved Electrolyte Model such as total system current, electrolyte thickness, electrolyte resistivity, and so on.

The **Enable Electrolyte Conductivity Submodel** option allows the ionic conductivity (or resistivity) of the electrolyte to change as a function of temperature. At the moment, there is one correlation that provides ionic conductivity (or resistivity) of the electrolyte as function of temperature.

$$\text{resistivity} = \frac{0.3685 + 0.002838e^{\frac{10300}{T}}}{100} \quad (6.1)$$

### Important

Note that this is valid only for temperatures ranging from 1073 K to 1373 K.

By turning off the **Enable Surface Energy Source** option, ANSYS Fluent excludes the heat addition due to electrochemistry and all the reversible processes. This option should be turned on at all times.

The **Enable Volumetric Energy Source** option includes the ohmic heating throughout the electrically conducting zones. You should keep this option turned off (to avoid slowing the convergence rates) until a certain rate of convergence for the potential field has been achieved, at which point, you should turn the option on manually. Note that this option is important so that the solution can account for the effects of the internal Ohmic heating.

The **Disable CO Electrochemistry** is enabled if there is carbon monoxide (CO) in the fuel line and if you do not want to include the CO in the electrochemistry.

Since the calculations are very sensitive to large current fluctuations early in the solution process, it is recommended that you use 0.3 or 0.4 for the **Current Under-Relaxation Factor** for a more effective solution.

The leakage current is the total amount of current due to the leakage of oxidizer to the fuel side (through the electrolyte) and the electric current across the electrolyte due to any short circuit. You can specify a value for the leakage current under **Leakage Current Density**.

If the leakage current density is temperature-dependent, you can specify your own temperature-dependent implementation of the leakage current density by performing the following steps:

1. Make the required changes to `Leakage_Current_Density (real T)` which is a real function in the user-modifiable source code, `constit.c`.
2. Compile `constit.c` and make your own `sofc` UDF library.

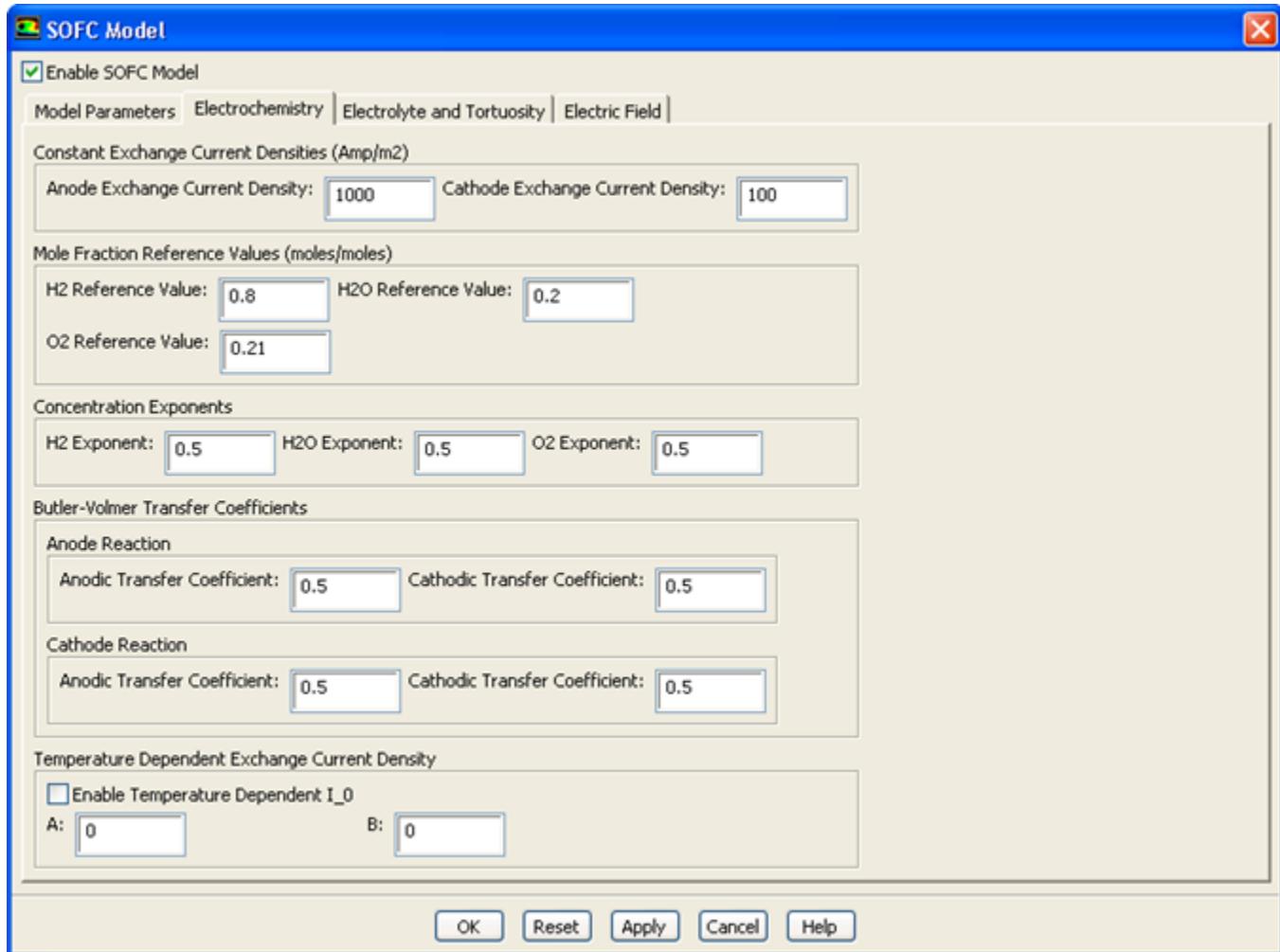
For more information, see [Compiling the Customized Solid Oxide Fuel Cell With Unresolved Electrolyte Source Code \(p. 352\)](#).

The **Converge to Specified System Voltage** is used when you want to specify a system voltage instead of system current as an input.

The **Set Individual Electrical Boundary Condition from Boundary Conditions Task Page** option, when enabled, allows you to directly specify the current density or voltage for each individual current-collecting boundary using the ANSYS Fluent **Boundary Conditions** task page. This allows you to apply multiple current types or multiple voltage types (either constant or variable) to your boundary conditions.

## 6.6. Setting Up the Electrochemistry Parameters

You can specify electrochemistry settings for the SOFC With Unresolved Electrolyte Model using the **Electrochemistry** tab in the **SOFC Model** dialog box.

**Figure 6.3: The Electrochemistry Tab in the SOFC Model Dialog Box**

In the **Electrochemistry** tab, you can set the anode and cathode exchange current density, the anode and cathode mole fraction reference values, the concentration exponents, the Butler-Volmer transfer coefficients, and the temperature-dependent exchange current density.

You can specify a value for the exchange current density at the anode (the default value is 1000 *Amps/m<sup>2</sup>*) using the **Anode Exchange Current Density** field. Likewise, you can specify a value for the exchange current density at the cathode (the default value is 100 *Amps/m<sup>2</sup>*) using the **Cathode Exchange Current Density** field.

You can also specify **Mole Fraction Reference Values** (Equation 5.18 (p. 332)-Equation 5.20 (p. 332)) for the fuel cell reactants in the **Electrochemistry** tab. By default, the reference value for  $H_2$  is 0.8, the reference value for  $O_2$  is 0.21, and the reference value for  $H_2O$  is 0.2.

The Butler-Volmer transfer coefficients can be set in the **Electrochemistry** tab as well. These coefficients are the  $\alpha_a$  and  $\alpha_c$  from Equation 5.15 (p. 331) for both the anode and the cathode reactions.

$$i = i_0 \left[ e^{\frac{\alpha_a n(\phi - \phi_0)F}{RT}} - e^{-\frac{\alpha_c n(\phi - \phi_0)F}{RT}} \right] \quad (6.2)$$

Remember that  $\alpha$  has anodic and cathodic values at *both* the cathode and the anode. By default, the value of  $\alpha$  is set to 0.5 because of the nearly universal assumption that there is a symmetric balance between the forward and backward reactions. In most cases, these default values will be sufficient.

If you find yourself changing the Butler-Volmer transfer coefficients, or if you have some other rate-limiting reaction in your fuel cell simulation, you may also want to consider changing the exponents for the stoichiometric coefficients for the fuel cell reactants. These exponents can be specified in the **Electrochemistry** tab. By default, the exponent values for  $H_2$ ,  $O_2$ , and  $H_2O$  are 0.5.

The **Enable Temperature Dependant I<sub>0</sub>** option allows the exchange current density to change as a function of temperature in an exponential fashion

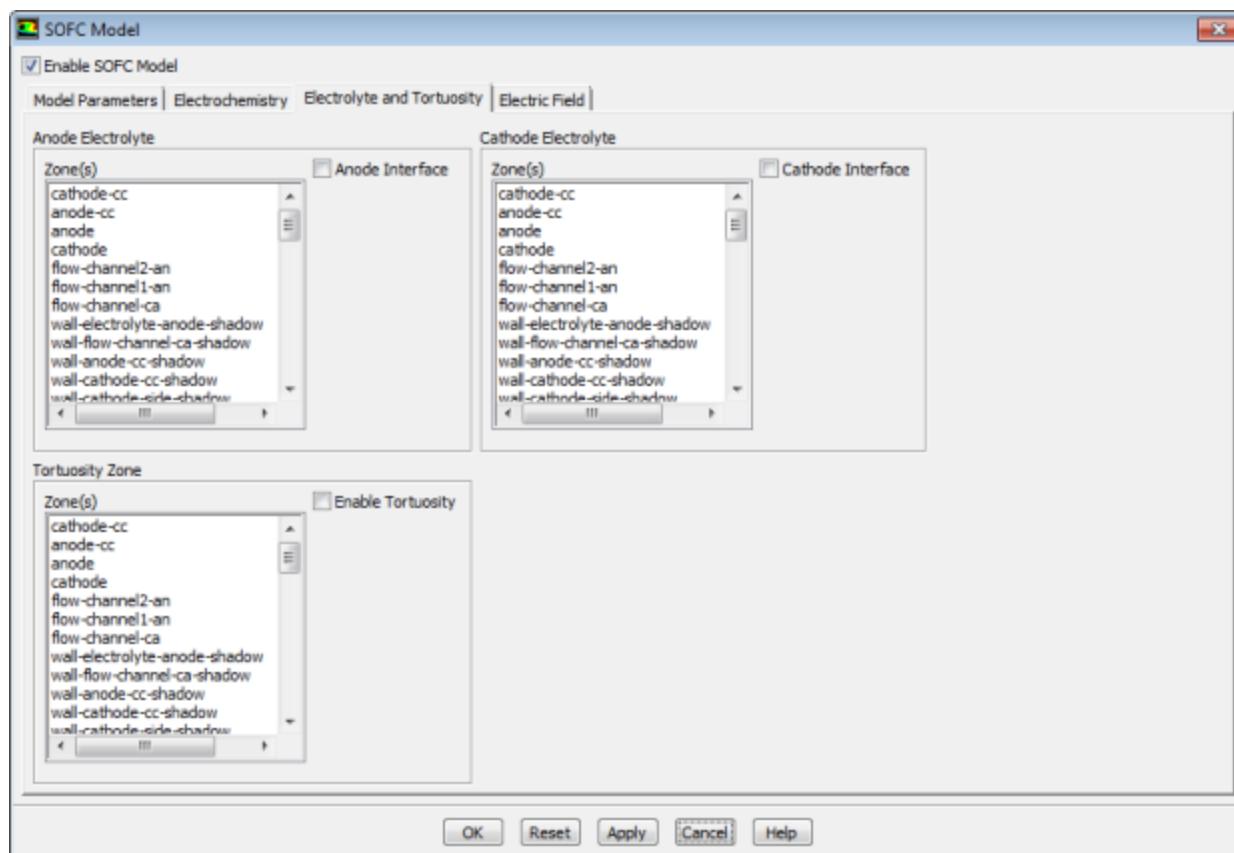
$$i_0 = A e^{-\frac{1}{BT}} \quad (6.3)$$

where you can provide the values for the constants  $A$  and  $B$ .

## 6.7. Setting Up the Electrode-Electrolyte Interfaces

You can apply specific settings for both your anode and your cathode interfaces as well as set tortuosity parameters using the ANSYS Fluent SOFC With Unresolved Electrolyte Model through the **Electrolyte** and **Tortuosity** tab in the **SOFC Model** dialog box.

**Figure 6.4: The Electrolyte and Tortuosity Tab in the SOFC Model Dialog Box**



1. Set up the anode electrode-electrolyte interface.
  - a. Enable the **Anode Interface** option.
  - b. Select the surface that represents the anode electrode interface with the electrolyte from the corresponding **Zone(s)** list
  - c. Click **Apply**.

2. In a similar manner, you can set up the cathode electrode-electrolyte interface.
3. Set up the tortuosity parameters.
  - a. Select the **Enable Tortuosity** option.
  - b. Select the an appropriate zone from the corresponding **Zone(s)** list
  - c. As required, assign tortuosity values to any porous zones in your simulation. In a porous zone, the mass diffusion coefficient is reduced as follows due to the porosity effect:

$$D_{eff} = \frac{porosity}{tortuosity} D \quad (6.4)$$

There typically are no standard means of measuring tortuosity as it must be either measured experimentally or tuned to match other experimental data. Tortuosity value may typically be in the range of 2 to 4, although you can use much higher values.

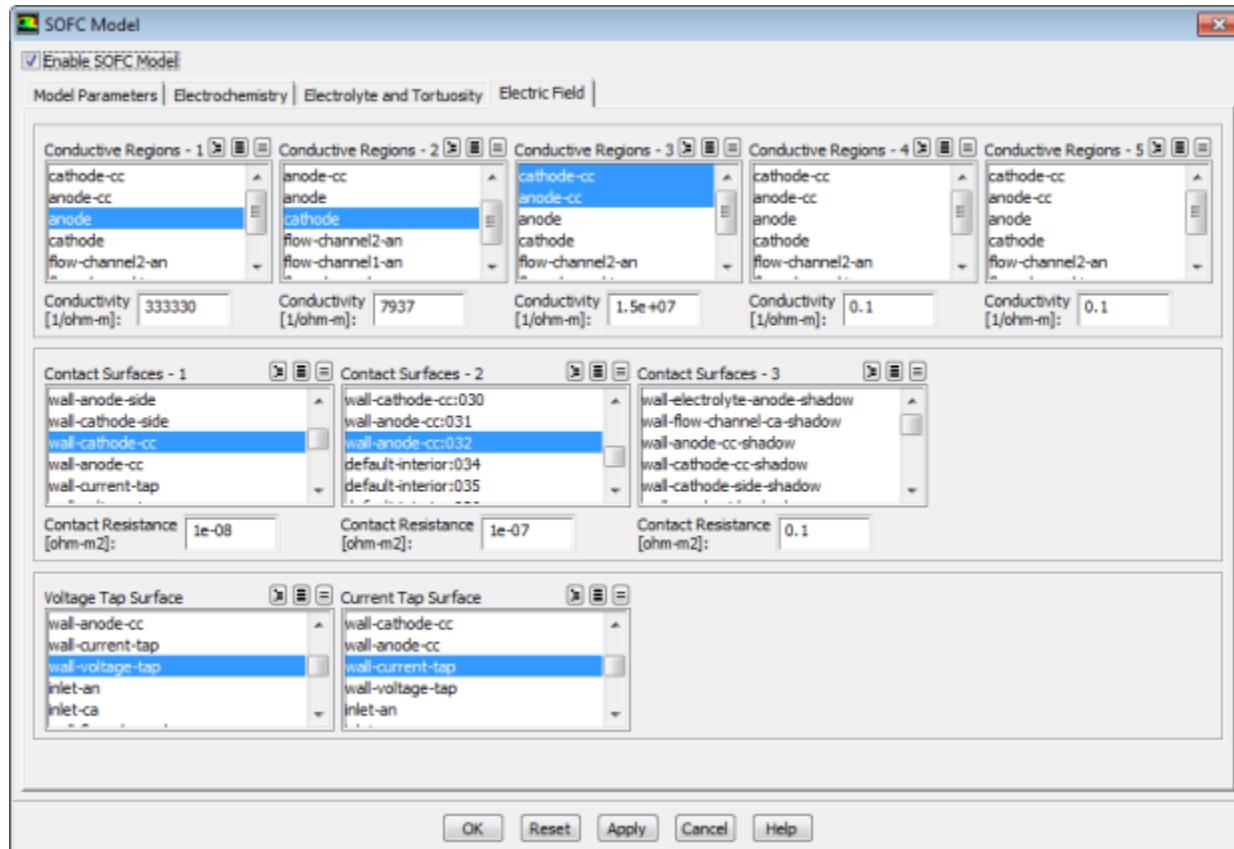
- d. Click **Apply**.

You can do this for as many zones as you need. When you are finished setting up the anode interface, click the **OK** button.

## 6.8. Setting Up the Electric Field Model Parameters

You can set up the details of the electric field model using the **Electric Field** tab in the **SOFC Model** dialog box.

**Figure 6.5: The Electric Field Tab in the SOFC Model Dialog Box**



Here, you can designate conductive regions and specify their conductivity, assign boundaries to be contact surfaces and specify the contact resistances, as well as specify which surfaces are grounded and which surfaces exhibit a current.

The **Voltage Tap Surface** is the surface that is grounded (that is,  $V=0$ ). The **Current Tap Surface** is the surface that current is being drawn from (that is,  $i$  is assigned a value).

## 6.9. User-Accessible Functions for the Solid Oxide Fuel Cell With Unresolved Electrolyte Model

You can directly incorporate your own formulations and data for the properties of the solid oxide fuel cell model using the `constit.c` source code file.

The following listing represents a description of the contents of the `constit.c` source code file:

```
real Nernst (real sp_a[], real sp_c[], real P_a, real P_c, real T_a, real T_c, real *DNDY)
```

Returns the Nernst potential using [Equation 5.9 \(p. 330\)](#).

```
real Activation (real Y[], real P, real T, real i, real i_0, real alpha_a, real alpha_b, int species, real *dA_di)
```

Returns the activation overpotential by solving [Equation 5.15 \(p. 331\)](#).

```
real Resistivity (real T)
```

Returns the electrolyte resistance (either a constant value, or computed using [Equation 6.1 \(p. 346\)](#)).

```
real Leakage_Current_Density(real T)
```

Returns the leakage current density specified in the **SOFC Model** dialog box and can be overwritten as a function of temperature.

```
real Solve_Butler_Volmer_NR (real i, real i0, real A, real B, real *detadi);
```

Returns the activation overpotential by solving [Equation 5.15 \(p. 331\)](#).

```
real CONDUCTIVITY_CELL(cell_t c, Thread *t)
```

Returns the cell conductivity value used in the **Electric Field** tab of the **SOFC Model** dialog box. This function can be overwritten to look up the conductive zone group ID (1 through 5). A conditional statement allows you to perform various tasks for respective conductive zones. For non-conductive zones, this function returns a value of 0.

```
real CONTACT_RESIST_FACE (face_t f, Thread *t)
```

Returns the contact resistance value used in the **Electric Field** tab of the **SOFC Model** dialog box. This function can be overwritten to look up the contact resistance group ID (1 through 3). A conditional statement allows you to perform various tasks for respective contact resistance surfaces. For non-contact resistance surfaces, this function returns a value of 0.

```
real h2_co_split_func(cell_t c_an, Thread *t_an)
```

Returns the  $H_2/CO$  split factor defined in [Equation 5.36 \(p. 335\)](#). This can be overwritten as needed to define your own split factor.

For more information, see [Compiling the Customized Solid Oxide Fuel Cell With Unresolved Electrolyte Source Code \(p. 352\)](#).

### 6.9.1. Compiling the Customized Solid Oxide Fuel Cell With Unresolved Electrolyte Source Code

## 6.9.1. Compiling the Customized Solid Oxide Fuel Cell With Unresolved Electrolyte Source Code

This section includes instructions on how to compile a customized Solid Oxide Fuel Cell With Unresolved Electrolyte user-defined module. Note that you can also refer to the file `INSTRUCTIONS-CLIENT` that comes with your distribution (see `addons/sofc`).

---

### Important

It is assumed that you have a basic familiarity with compiling user-defined functions (UDFs). For an introduction on how to compile UDFs, refer to the [Fluent Customization Manual](#).

---

You will first want to use a local copy of the `sofc` directory in the `addons` directory before you recompile the Solid Oxide Fuel Cell With Unresolved Electrolyte module.

### 6.9.1.1. Compiling the Customized Source Code Under Linux

1. Make a local copy of the `sofc` directory. Do not create a symbolic link.
- 

### Important

The custom version of the library must be named according to the convention used by ANSYS Fluent: for example, `sofc`.

---

2. Change directories to the `sofc/src` directory.
3. Make changes to the `constit.c` file.
4. Define the `FLUENT_ADDONS` environment variable to correspond to your customized version of the Solid Oxide Fuel Cell With Unresolved Electrolyte module.
5. Change directories to the `sofc/` directory.
6. Issue the following `make` command:

```
make FLUENT_INC=[ansys_inc/v170/fluent] FLUENT_ARCH=[arch]  
-f Makefile-client
```

where `your_arch` is `lnx86` on `LINUX`, or `ultra` on the Sun operating system, and so on.

The following example demonstrates the steps required to set up and run a customized version of the Fuel Cell and Electrolysis module that is located in a folder call `home/sample`:

1. Make a directory.

```
mkdir -p /home/sample
```

2. Copy the default addon library to this location.

```
cp -RH [ansys_inc/v170/fluent]/fluent17.0.0/addons/sofc  
/home/sample/sofc
```

3. Using a text editor, make the appropriate changes to the `constit.c` file located in `/home/sample/sofc/src/constit.c`
4. Build the library.

```
cd /home/sample/sofc
make FLUENT_INC=[ansys_inc/v170/fluent] FLUENT_ARCH=[arch]
-f Makefile-client
```

5. Set the `FLUENT_ADDONS` environment variable (using CSH, other shells will differ).

```
setenv FLUENT_ADDONS /home/sample
```

6. Start ANSYS Fluent and load the customized module using the text interface command.

### **6.9.1.2. Compiling the Customized Source Code Under Windows**

1. Open **Visual Studio.NET** at the DOS prompt.
2. Make sure that the `$FLUENT_INC` environment variable is correctly set to the current ANSYS Fluent installation directory (for example, `ANSYS Inc\v170\fluent`).
3. Make a local copy of the `sofc` folder. Do not create a shortcut.
4. Enter the `sofc\src` folder.
5. Make changes to the `constit.c` file.
6. Define the `FLUENT_ADDONS` environment variable to correspond to your customized version of the Solid Oxide Fuel Cell With Unresolved Electrolyte module.
7. Return to the `sofc` folder.
8. Issue the following command in the command window:

```
nmake /f makefile_master-client.nt
```

## **6.10. Using the Solid Oxide Fuel Cell With Unresolved Electrolyte Text User Interface**

All of the features for the Solid Oxide Fuel Cell With Unresolved Electrolyte Model that are available through the graphical user interface are also available through text user interface commands.

Once the fuel cell module is loaded (see [Loading the Solid Oxide Fuel Cell With Unresolved Electrolyte Module \(p. 337\)](#)), you can access the text user interface through the Console Window under `sofc-model`. A listing of the various text commands is as follows:

```
sofc-model/
SOFC model menu

enable-sofc-model?
Enable/disable SOFC model

model-parameters
Set model parameters
```

**electrochemistry**

Set electrochemistry parameters

**anode-interface**

Set fuel cell anode interface

**cathode-interface**

Set fuel cell cathode interface

**tortuosity-interface**

Set fuel cell tortuosity interface

**electric-field-model/**

Electric field model

**voltage-tap**

Set voltage tap surface

**current-tap**

Set current tap surface

**conductive-regions**

Set conductive regions

**contact-resistance-regions**

Set contact resistance regions

---

# Bibliography

- [1] *Fuel Cell Handbook (5th Edition)*. E G & G Services and Parsons Inc. and Science Applications International Corporation. 2000.
- [2] J.C. Barrett and C.F. Clement. "Growth rates for liquid drops". *J. Aerosol. Sci.*. 19(2). 223-242. 1988.
- [3] Brandon M. Eaton. "One-dimensional, Transient Model of Heat, Mass, and Charge Transfer in a Proton Exchange Membrane". *M.S. Thesis*. 2001.
- [4] A. A. Kulikovsky, J. Divisek, and A. A. Kornyshev. "Modeling the Cathode Compartment of Polymer Electrolyte Fuel Cells: Dead and Active Reaction Zones". *Journal, Electrochemical Society*. 146(11). 3981–3991. 1999.
- [5] S. Mazumder and J. V. Cole . "Rigorous 3-D Mathematical Modeling of PEM Fuel Cells II. Model Predictions with Liquid Water Transport". *Journal, Electrochemical Society*. 150(11). A1510–1517. 2003.
- [6] J. H. Nam and M. Karviany. "Effective Diffusivity and Water-Saturation Distribution in Single- and Two-layer PEMFC Diffusion Medium". *Int. Journal Heat Mass Transfer*. 2003.
- [7] J. S. Newman. *Electrochemical Systems*. Prentice Hall. Englewood Cliffs, New Jersey 1973.
- [8] T. V. Nguyen. "Modeling Two-Phase Flow in the Porous Electrodes of Proton Exchange Membrane Fuel Cells Using the Interdigitated Flow Fields". *Tutorials in Electrochemical Engineering Mathematical Modeling*. 99(14). 222–241. 1999.
- [9] U. Pasaogullari and C-Y. Wang. "Two-Phase transport and the role of micro-porous layer in polymer electrolyte fuel cells". *Electrochimica Acta*. 49(25). 4359-4369. 2004.
- [10] H. Scholz. "Modellierung und Untersuchung des Wärme- und Stofftransports und von Flutungsphänomenen in Niedertemperatur-PEM-Brennstoffzellen". *PhD Thesis*. 2015.
- [11] T. E. Springer, T. A. Zawodzinski, and S. Gottesfeld. "Polymer Electrolyte Fuel Cell Model". *Journal, Electrochemical Society*. 138. 2334–2342. 1991.
- [12] Sukkee Um, C. Y. Wang, and K. S. Chen. "Computational Fluid Dynamics Modeling of Proton Exchange Membrane Fuel Cells". *Journal, Electrochemical Society*. 147(12). 4485–4493. 2000.
- [13] H. Wu, X. Li and P. Berg. "On the modeling of water transport in polymer electrolyte membrane fuel cells". *Electrochimica Acta*. 54(27). 6913-6927. 2009.



---

## **Part VI: ANSYS Fluent Magnetohydrodynamics (MHD) Module**

[Using This Manual \(p. cclix\)](#)

[1. Introduction \(p. 361\)](#)

[2. Magnetohydrodynamic Model Theory \(p. 363\)](#)

[3. Implementation \(p. 367\)](#)

[4. Using the ANSYS Fluent MHD Module \(p. 369\)](#)

[Appendix A. Guidelines For Using the ANSYS Fluent MHD Model \(p. 383\)](#)

[Appendix B. Definitions of the Magnetic Field \(p. 387\)](#)

[Appendix C. External Magnetic Field Data Format \(p. 389\)](#)

[Appendix D. MHD Module Text Commands \(p. 391\)](#)

[Bibliography \(p. 393\)](#)

---

---

---

# Using This Manual

---

## 1. The Contents of This Manual

The ANSYS Fluent Magnetohydrodynamics (MHD) Module Manual tells you what you need to know to model magnetohydrodynamics with ANSYS Fluent. In this manual, you will find background information pertaining to the model, a theoretical discussion of the model used in ANSYS Fluent, and a description of using the model for your CFD simulations.

This part is divided into the following chapters:

- [Introduction \(p. 361\)](#)
- [Magnetohydrodynamic Model Theory \(p. 363\)](#)
- [Implementation \(p. 367\)](#)
- [Using the ANSYS Fluent MHD Module \(p. 369\)](#)
- [Appendix A \(p. 383\)](#)
- [Appendix B \(p. 387\)](#)
- [Appendix C \(p. 389\)](#)
- [Appendix D \(p. 391\)](#)
- [Bibliography \(p. 393\)](#)



---

---

## Chapter 1: Introduction

---

The Magnetohydrodynamics (MHD) module is provided as an add-on module with the standard ANSYS Fluent licensed software.

Magnetohydrodynamics refers to the interaction between an applied electromagnetic field and a flowing, electrically-conductive fluid. The ANSYS Fluent MHD model enables you to analyze the behavior of electrically conducting fluid flow under the influence of constant (DC) or oscillating (AC) electromagnetic fields. The externally-imposed magnetic field may be generated either by selecting simple built-in functions or by importing a user-supplied data file. For multiphase flows, the MHD model is compatible with both the discrete phase model (DPM), the volume-of-fluid (VOF) and Eulerian mixture approaches in ANSYS Fluent, including the effects of a discrete phase on the electrical conductivity of the mixture.

This document describes the ANSYS Fluent MHD model. [Magnetohydrodynamic Model Theory \(p. 363\)](#) provides theoretical background information. [Implementation \(p. 367\)](#) summarizes the UDF-based software implementation. Instructions for getting started with the model are provided in [Using the ANSYS Fluent MHD Module \(p. 369\)](#). [Appendix A \(p. 383\)](#) provides a condensed overview on how to use the MHD model, while [Appendix B \(p. 387\)](#) contains definitions for the magnetic field, [Appendix C \(p. 389\)](#) describes the external magnetic field data format, and [Appendix D \(p. 391\)](#) lists the text commands in the MHD model.



---

## Chapter 2: Magnetohydrodynamic Model Theory

---

This chapter presents an overview of the theory and the governing equations for the mathematical models used in ANSYS Fluent to predict flow in an electromagnetic field.

### 2.1. Introduction

#### 2.2. Magnetic Induction Method

#### 2.3. Electric Potential Method

## 2.1. Introduction

The coupling between the fluid flow field and the magnetic field can be understood on the basis of two fundamental effects: the induction of electric current due to the movement of conducting material in a magnetic field, and the effect of Lorentz force as the result of electric current and magnetic field interaction. In general, the induced electric current and the Lorentz force tend to oppose the mechanisms that create them. Movements that lead to electromagnetic induction are therefore systematically braked by the resulting Lorentz force. Electric induction can also occur in the presence of a time-varying magnetic field. The effect is the stirring of fluid movement by the Lorentz force.

Electromagnetic fields are described by Maxwell's equations:

$$\nabla \cdot \vec{B} = 0 \quad (2.1)$$

$$\nabla \times \vec{E} = -\frac{\partial \vec{B}}{\partial t} \quad (2.2)$$

$$\nabla \cdot \vec{D} = q \quad (2.3)$$

$$\nabla \times \vec{H} = \vec{j} + \frac{\partial \vec{D}}{\partial t} \quad (2.4)$$

where  $\vec{B}$  (Tesla) and  $\vec{E}$  (V/m) are the magnetic and electric fields, respectively, and  $\vec{H}$  and  $\vec{D}$  are the induction fields for the magnetic and electric fields, respectively.  $q$  ( $C/m^3$ ) is the electric charge density, and  $\vec{j}$  ( $A/m^2$ ) is the electric current density vector.

The induction fields  $\vec{H}$  and  $\vec{D}$  are defined as:

$$\vec{H} = \frac{1}{\mu} \vec{B} \quad (2.5)$$

$$\vec{D} = \epsilon \vec{E} \quad (2.6)$$

where  $\mu$  and  $\epsilon$  are the magnetic permeability and the electric permittivity, respectively. For sufficiently conducting media such as liquid metals, the electric charge density  $q$  and the displacement current  $\frac{\partial \vec{D}}{\partial t}$  are customarily neglected [1] (p. 393).

In studying the interaction between flow field and electromagnetic field, it is critical to know the current density  $\vec{j}$  due to induction. Generally, two approaches may be used to evaluate the current density. One is through the solution of a magnetic induction equation; the other is through solving an electric potential equation.

## 2.2. Magnetic Induction Method

In the first approach, the magnetic induction equation is derived from Ohm's law and Maxwell's equation. The equation provides the coupling between the flow field and the magnetic field.

In general, Ohm's law that defines the current density is given by:

$$\vec{j} = \sigma \vec{E} \quad (2.7)$$

where  $\sigma$  is the electrical conductivity of the media. For fluid velocity field  $\vec{U}$  in a magnetic field  $\vec{B}$ , Ohm's law takes the form:

$$\vec{j} = \sigma (\vec{E} + \vec{U} \times \vec{B}) \quad (2.8)$$

From Ohm's law and Maxwell's equation, the induction equation can be derived as:

$$\frac{\partial \vec{B}}{\partial t} + (\vec{U} \cdot \nabla) \vec{B} = \frac{1}{\mu \sigma} \nabla^2 \vec{B} + (\vec{B} \cdot \nabla) \vec{U} \quad (2.9)$$

From the solved magnetic field  $\vec{B}$ , the current density  $\vec{j}$  can be calculated using Ampere's relation as:

$$\vec{j} = \frac{1}{\mu} \nabla \times \vec{B} \quad (2.10)$$

Generally, the magnetic field  $\vec{B}$  in a MHD problem can be decomposed into the externally imposed field  $\vec{B}_0$  and the induced field  $\vec{b}$  due to fluid motion. Only the induced field  $\vec{b}$  must be solved.

From Maxwell's equations, the imposed field  $\vec{B}_0$  satisfies the following equation:

$$\nabla^2 \vec{B}_0 - \mu \sigma' \frac{\partial \vec{B}_0}{\partial t} = 0 \quad (2.11)$$

where  $\sigma'$  is the electrical conductivity of the media in which field  $\vec{B}_0$  is generated. Two cases need to be considered.

For more information, see the following sections:

- [2.2.1. Case 1: Externally Imposed Magnetic Field Generated in Non-conducting Media](#)
- [2.2.2. Case 2: Externally Imposed Magnetic Field Generated in Conducting Media](#)

### 2.2.1. Case 1: Externally Imposed Magnetic Field Generated in Non-conducting Media

In this case the imposed field  $\vec{B}_0$  satisfies the following conditions:

$$\nabla \times \vec{B}_0 = 0 \quad (2.12)$$

$$\nabla^2 \vec{B}_0 = 0 \quad (2.13)$$

With  $\vec{B} = \vec{B}_0 + \vec{b}$ , the induction equation (Equation 2.9 (p. 364)) can be written as:

$$\frac{\partial \vec{b}}{\partial t} + (\vec{U} \cdot \nabla) \vec{b} = \frac{1}{\mu \sigma} \nabla^2 \vec{b} + ((\vec{B}_0 + \vec{b}) \cdot \nabla) \vec{U} - (\vec{U} \cdot \nabla) \vec{B}_0 - \frac{\partial \vec{B}_0}{\partial t} \quad (2.14)$$

The current density is given by:

$$\vec{j} = \frac{1}{\mu} \nabla \times \vec{b} \quad (2.15)$$

## 2.2.2. Case 2: Externally Imposed Magnetic Field Generated in Conducting Media

In this case the conditions given in [Equation 2.12 \(p. 364\)](#) and [Equation 2.13 \(p. 364\)](#) are not true. Assuming that the electrical conductivity of the media in which field  $\vec{B}_0$  is generated is the same as that of the flow, that is  $\sigma'=\sigma$ , from [Equation 2.9 \(p. 364\)](#) and [Equation 2.11 \(p. 364\)](#) the induction equation can be written as:

$$\frac{\partial \vec{b}}{\partial t} + (\vec{U} \cdot \nabla) \vec{b} = \frac{1}{\mu\sigma} \nabla^2 \vec{b} + ((\vec{B}_0 + \vec{b}) \cdot \nabla) \vec{U} - (\vec{U} \cdot \nabla) \vec{B}_0 \quad (2.16)$$

and the current density is given by:

$$\vec{j} = \frac{1}{\mu} \nabla \times (\vec{B}_0 + \vec{b}) \quad (2.17)$$

For the induction equation [Equation 2.14 \(p. 364\)](#) or [Equation 2.16 \(p. 365\)](#), the boundary conditions for the induced field are given by:

$$\vec{b} = \begin{Bmatrix} b_n & b_{t1} & b_{t2} \end{Bmatrix}^T = \vec{b}^* \quad (2.18)$$

where the subscripts denote the normal and tangential components of the field and  $\vec{b}^*$  is specified by you. For an electrically insulating boundary, as  $j_n=0$  at the boundary, from Ampere's relation one has  $b_{t1}=b_{t2}=0$  at the boundary.

## 2.3. Electric Potential Method

The second approach for the current density is to solve the electric potential equation and calculate the current density using Ohm's law. In general, the electric field  $\vec{E}$  can be expressed as:

$$\vec{E} = -\nabla \varphi - \frac{\partial \vec{A}}{\partial t} \quad (2.19)$$

where  $\varphi$  and  $\vec{A}$  are the scalar potential and the vector potential, respectively. For a static field and assuming  $\vec{b} \ll \vec{B}_0$ , Ohm's law given in [Equation 2.8 \(p. 364\)](#) can be written as:

$$\vec{j} = \sigma (-\nabla \varphi + (\vec{U} \times \vec{B}_0)) \quad (2.20)$$

For sufficiently conducting media, the principle of conservation of electric charge gives:

$$\nabla \cdot \vec{j} = 0 \quad (2.21)$$

The electric potential equation is therefore given by:

$$\nabla^2 \varphi = \nabla \cdot (\vec{U} \times \vec{B}_0) \quad (2.22)$$

The boundary condition for the electric potential  $\varphi$  is given by:

$$\frac{\partial \varphi}{\partial n} = (\vec{U} \times \vec{B}_0)_{boundary} \cdot \vec{n} \quad (2.23)$$

for an insulating boundary, where  $\vec{n}$  is the unit vector normal to the boundary, and

$$\varphi = \varphi_0 \quad (2.24)$$

for a conducting boundary, where  $\varphi_0$  is the specified potential at the boundary. The current density can then be calculated from [Equation 2.20 \(p. 365\)](#).

With the knowledge of the induced electric current, the MHD coupling is achieved by introducing additional source terms to the fluid momentum equation and energy equation. For the fluid momentum equation, the additional source term is the Lorentz force given by:

$$\vec{F} = \vec{j} \times \vec{B} \quad (2.25)$$

which has units of N/m N/m<sup>3</sup> in the SI system. For the energy equation, the additional source term is the Joule heating rate given by:

$$Q = \frac{1}{\sigma} \vec{j} \cdot \vec{j} \quad (2.26)$$

which has units of W/m<sup>3</sup>.

For charged particles in an electromagnetic field, the Lorentz force acting on the particle is given by:

$$\vec{F}_p = q(\vec{E} + \vec{v}_p \times \vec{B}) \quad (2.27)$$

where  $q$  is the particle charge density (Coulomb/m<sup>3</sup>) and  $v_p$  is the particle velocity. The force  $\vec{F}_p$  has units of N/m<sup>3</sup>.

For multiphase flows, assuming that the electric surface current at the interface between phases can be ignored, the electric conductivity for the mixture is given by:

$$\sigma_m = \sum_i \sigma_i v_i \quad (2.28)$$

where  $\sigma_i$  and  $v_i$  are respectively the electric conductivity and volume fraction of phase  $i$ .  $\sigma_m$  is used in solving the induction equations.

---

## Chapter 3: Implementation

---

The MHD model is implemented using user-defined functions (UDF) as an ANSYS Fluent add-on module, which is loaded into ANSYS Fluent at run time. The model is accessed through a number of UDF schemes. The magnetic induction equation given by [Equation 2.14 \(p. 364\)](#) or [Equation 2.16 \(p. 365\)](#) and the electric potential equation given by [Equation 2.22 \(p. 365\)](#) are solved through user-defined scalar (UDS) transport equations. Other model-related variables such as the external magnetic field data, current density, Lorentz force and Joule heat are stored as user-defined memory (UDM) variables. The MHD model setup and parameters are entered using the **MHD Model** graphical user interface (GUI) dialog box and a set of text user interface (TUI) commands described in [Using the ANSYS Fluent MHD Module \(p. 369\)](#). Detailed information can be found in the following sections:

- [3.1. Solving Magnetic Induction and Electric Potential Equations](#)
- [3.2. Calculation of MHD Variables](#)
- [3.3. MHD Interaction with Fluid Flows](#)
- [3.4. MHD Interaction with Discrete Phase Model](#)
- [3.5. General User-Defined Functions](#)

### 3.1. Solving Magnetic Induction and Electric Potential Equations

The magnetic induction equation and the electric potential equations are solved through user-defined scalar transport equations. For the magnetic induction equation a set of 2 or 3 scalar equations are solved, each representing a Cartesian component of the induced magnetic field vector in a 2D or 3D case. For the electric potential equation a single scalar equation is solved.

The convection and the diffusion terms of the scalar equations are defined using user functions `DEFINE_UDS_FLUX(mhd_flux, ..., ns)` and `DEFINE_DIFFUSIVITY (mhd_magnetic_diffusivity, ..., ns)` respectively. The user-defined scalar equation is identified by the scalar index `ns`.

The source terms to the induction equations and the potential equation are implemented using user function `DEFINE_SOURCE(mhd_mag_source, ..., eqn)` and `DEFINE_SOURCE (mhd_phi_source, ..., eqn)` respectively, where `eqn` identifies the scalar equations.

For transient cases, the additional unsteady source term is introduced through the user function `DEFINE_UDS_UNSTEADY(mhd_unsteady_source, ..., ns)`, where `ns` identifies the scalar being solved.

The induction and potential equations can also be solved in solid zones, in which case the fluid velocity terms in the equations are not considered. For multiphase flows, the MHD equations are solved in the mixture domain only.

The wall boundary conditions are implemented through user profile functions (`DEFINE_PROFILE(mhd_bc_...)`), and are applied to the Cartesian components of the induced magnetic field vector or to the electric potential. For external wall boundaries, three types of boundary conditions (electrically insulating, conducting, and "thin wall"), can be applied. The 'thin wall' type boundary refers to an external wall where a 1D magnetic or electric potential diffusion normal to the boundary is assumed,

and the wall material and the thickness are specified for the boundary. For internal wall boundaries, that is the boundaries between fluid/solid or solid/solid zones, a coupled boundary condition is applied.

## 3.2. Calculation of MHD Variables

Apart from the Cartesian components of the magnetic field vectors and the electric potential function, which are stored as user-defined scalars, other MHD-related variables include the induced electric current density vector, induced electric field vector, the Lorentz force vector and Joule heat. These variables are stored in user-defined memory locations. Updating of MHD variables is accessed through the user function `DEFINE_ADJUST(mhd_adjust, ...)`. The variables are updated at the start of each iteration using the solved induced magnetic field from the previous iteration.

## 3.3. MHD Interaction with Fluid Flows

Additional source terms due to the magnetic induction are added to the flow momentum and energy equations as user defined source terms. For the momentum equation, user function `DEFINE_SOURCE(mhd_mom_source, ..., eqn)` is used to introduce the Lorentz force to the equation, where `eqn` identifies the Cartesian component of the fluid momentum. For the energy equation, the additional source due to Joule heating is added through user function `DEFINE_SOURCE(mhd_energy_source, ..., eqn)`, where `eqn` is the energy equation index.

## 3.4. MHD Interaction with Discrete Phase Model

In discrete phase modeling, the Lorentz force acting on charged particles is introduced through the user function `DEFINE_DPM_BODY_FORCE(mhd_dpm_force, ...)`. User function `DEFINE_DPM_SOURCE(mhd_dpm_source, ...)` is used to update the volume fraction of the discrete phase inside a fluid cell and the volume-weighted electric conductivity of the discrete phase.

## 3.5. General User-Defined Functions

Several general UDFs are used as part of the MHD model implementation.

- `DEFINE_INIT(mhd_init, ...)` is an initialization function called during the general case initialization to set up the external magnetic field and initialize MHD model parameters and variables.
- `DEFINE_ADJUST(mhd_adjust, ...)` is called at the start of each iteration. It is used to adjust the magnetic boundary conditions and update MHD related variables and properties.

---

## Chapter 4: Using the ANSYS Fluent MHD Module

---

This chapter provides basic instructions to install the magnetohydrodynamics (MHD) module and solve MHD problems in ANSYS Fluent. It assumes that you are already familiar with standard ANSYS Fluent features, including the user-defined function procedures described in the [Fluent Customization Manual](#). [Appendix A \(p. 383\)](#) also outlines the general procedure for using the MHD model. This chapter describes the following:

- 4.1. MHD Module Installation
- 4.2. Loading the MHD Module
- 4.3. MHD Model Setup
- 4.4. MHD Solution and Postprocessing
- 4.5. Limitations

### 4.1. MHD Module Installation

The MHD module is provided as an add-on module with the standard ANSYS Fluent licensed software. A special license is required to use the MHD module. The module is installed with the standard installation of ANSYS Fluent in a directory called `addons/mhd` in your installation area. The MHD module consists of a UDF library and a pre-compiled scheme library, which must be loaded and activated before calculations can be performed.

### 4.2. Loading the MHD Module

The MHD module is loaded into ANSYS Fluent through the text user interface (TUI). The module can only be loaded when a valid ANSYS Fluent case file has been set or read. The text command to load the module is

`define → models → addon-module.`

A list of ANSYS Fluent add-on modules is displayed:

```
> /define/models/addon-module
Fluent  Addon Modules:
 0. None
 1. MHD Model
 2. Fiber Model
 3. Fuel Cell and Electrolysis Model
 4. SOFC Model with Unresolved Electrolyte
 5. Population Balance Model
 6. Adjoint Solver
 7. Battery Module
 8. MSMD Battery Model
 9. PEM Fuel Cell Model
Enter Module Number: [0] 1
```

Select the MHD model by entering the module number 1. During the loading process a scheme library containing the graphical and text user interface, and a UDF library containing a set of user defined functions are loaded into ANSYS Fluent. A message **Addon Module: mhd...loaded!** is displayed at the end of the loading process.

The basic set up of the MHD model is performed automatically when the MHD module is loaded successfully. The set up includes:

- Selecting the default MHD method
- Allocating the required number of user-defined scalars and memory locations naming of:
  - User-defined scalars and memory locations
  - All UDF Hooks for MHD initialization and adjustment
  - MHD equation flux and unsteady terms
  - Source terms for the MHD equations
  - Additional source terms for the fluid momentum and energy equations
  - Default MHD boundary conditions for external and internal boundaries
  - A default set of model parameters
- DPM related functions are also set, if the DPM option has been selected in the ANSYS Fluent case set up.

The MHD module set up is saved with the ANSYS Fluent case file. The module is loaded automatically when the case file is subsequently read into ANSYS Fluent. Note that in the saved case file, the MHD module is saved with the absolute path. Therefore, if the locations of the MHD module installation or the saved case file are changed, ANSYS Fluent will not be able to load the module when the case file is subsequently read.

To unload the previously saved MHD module library, open the **UDF Library Manager** dialog box

 **Parameters & Customization** → **User Defined Functions**  **Manage...**

and reload the module as described above. Note that the previously saved MHD model setup and parameters are preserved.

## 4.3. MHD Model Setup

Following the loading of the MHD module, you can access the **MHD Model** dialog box using

 **Setup** → **Models** → **MHD Model**  **Edit...**

or using the text command

`define → models → mhd-model`

Both the **MHD Model** dialog box and TUI commands are designed for the following tasks:

- Enable/disable the MHD model.
- Select the MHD method.
- Apply an external magnetic field.

- Set boundary conditions.
- Set solution control parameters.

Operations of these tasks through the **MHD Model** dialog box are described in the following sections. The set of MHD text commands are listed in [Appendix D \(p. 391\)](#).

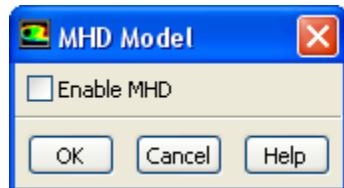
For more information, see the following sections:

- 4.3.1. Enabling the MHD Model
- 4.3.2. Selecting an MHD Method
- 4.3.3. Applying an External Magnetic Field
- 4.3.4. Setting Up Boundary Conditions
- 4.3.5. Solution Controls

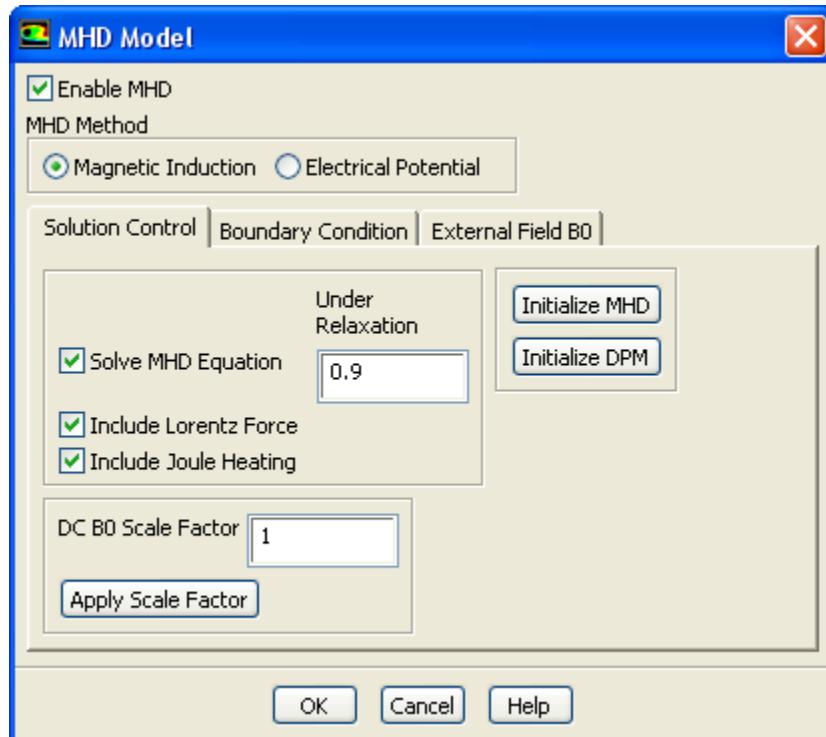
### 4.3.1. Enabling the MHD Model

If the MHD model is not enabled after the MHD module is loaded for the first time, you can enable it by selecting **Enable MHD** in the **MHD Model** dialog box, shown in [Figure 4.1: Enabling the MHD Model Dialog Box \(p. 371\)](#). The dialog box expands to its full size when the model is enabled, as shown in [Figure 4.2: The MHD Model Dialog Box \(p. 371\)](#).

**Figure 4.1: Enabling the MHD Model Dialog Box**



**Figure 4.2: The MHD Model Dialog Box**



### 4.3.2. Selecting an MHD Method

The method used for MHD calculation can be selected under **MHD Method** in the **MHD Model** dialog box. The two methods, **Magnetic Induction** and **Electrical Potential**, are described in [Magnetic Induction Method \(p. 364\)](#) and [Electric Potential Method \(p. 365\)](#).

For the Magnetic Induction method, 2 or 3 user-defined scalars are allocated for the solution of the induced magnetic field in 2D or 3D cases. The scalars are listed as  $B_x$ ,  $B_y$  and  $B_z$  representing the Cartesian components of the induced magnetic field vector. The unit for the scalar is Tesla.

For the Electrical Potential method, 1 user-defined scalar is solved for the electric potential field. The scalar is listed as  $\varphi$  and has the unit of Volt.

[Table 4.1: User-Defined Scalars in MHD Model \(p. 372\)](#) lists the user-defined scalars used by the two methods.

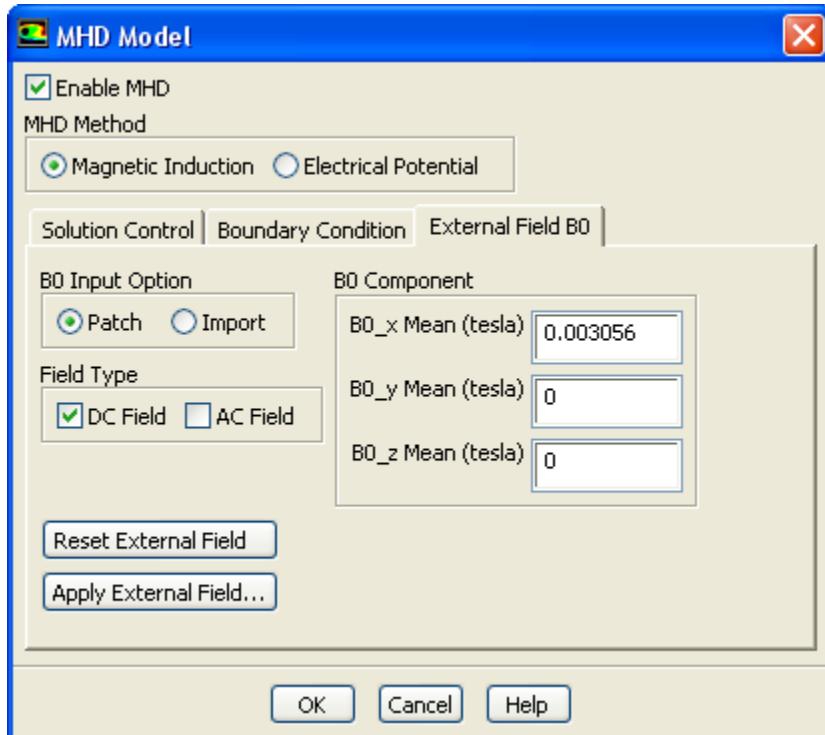
**Table 4.1: User-Defined Scalars in MHD Model**

Method	Scalar	Name	Unit	Description
Induction	Scalar-0	$B_x$	Tesla	X component of induced magnetic field ( $b_x$ )
	Scalar-1	$B_y$	Tesla	Y component of induced magnetic field ( $b_y$ )
	Scalar-2 (3-D)	$B_z$	Tesla	Z component of induced magnetic field ( $b_z$ )
Potential	Scalar-0	Phi	Volt	Electric potential ( $\varphi$ )

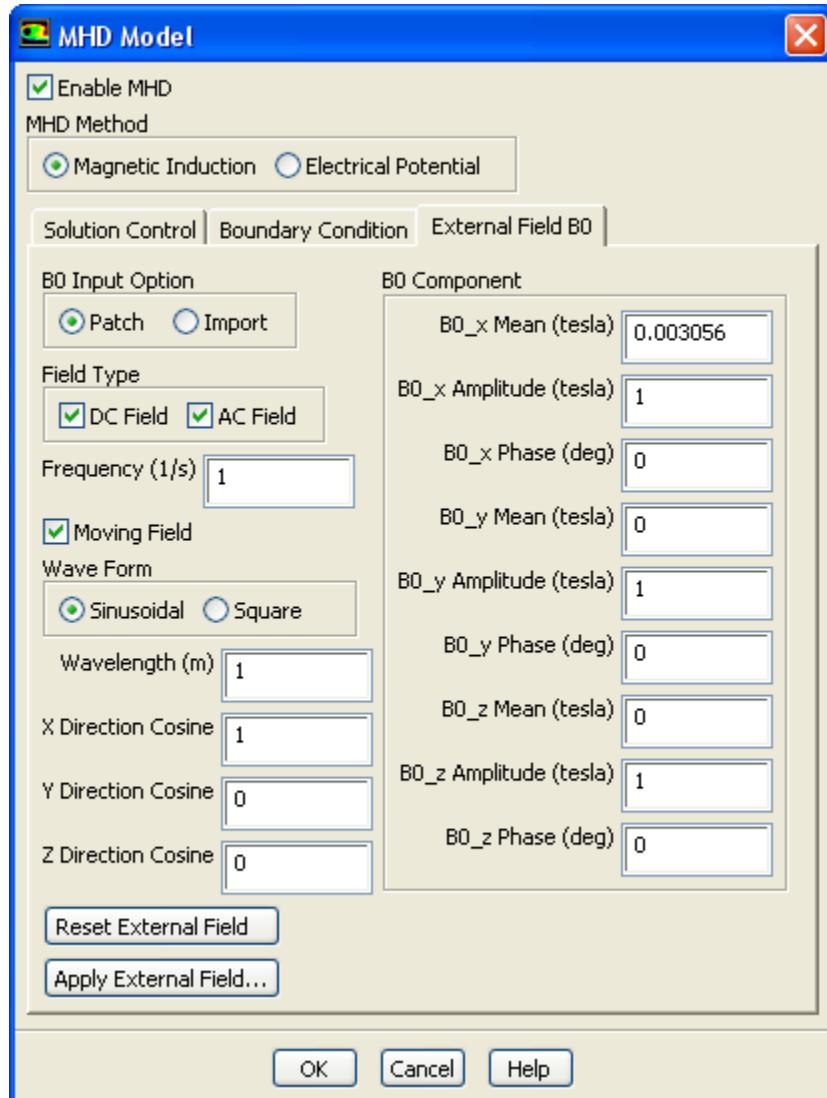
### 4.3.3. Applying an External Magnetic Field

Application of an external magnetic field to the computation domain is done under the **External Field B0** tab in the **MHD Model** dialog box, as shown in [Figure 4.3: The MHD Model Dialog Box for Patching an External Magnetic Field \(p. 373\)](#). Two **B0 Input Options** are available for setting up the external magnetic field. One option is to **Patch** the computational domain with a constant (**DC Field**) and/or varying (**AC Field**) type. The other option is to **Import** the field data from a magnetic data file that you provide.

With the **Patch** option enabled, the AC field can be expressed as a function of time (specified by **Frequency**), and space (specified by wavelength, propagation direction and initial phase offset). The space components are set under the **B0 Component**, as in [Figure 4.3: The MHD Model Dialog Box for Patching an External Magnetic Field \(p. 373\)](#).

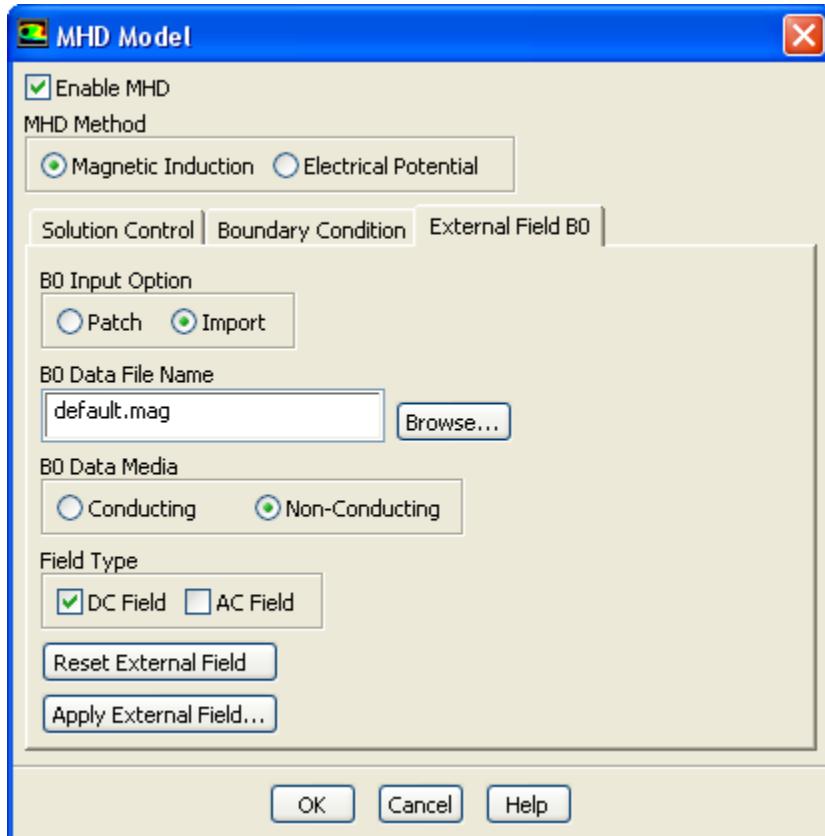
**Figure 4.3: The MHD Model Dialog Box for Patching an External Magnetic Field**

You can also specify a **Moving Field** with a wave form that is either a sinusoidal or a square wave function ([Figure 4.4: The MHD Model Dialog Box for Specifying a Moving Field \(p. 374\)](#)). Definitions for the sinusoidal and square wave forms of patched magnetic fields are provided in [Appendix B \(p. 387\)](#).

**Figure 4.4: The MHD Model Dialog Box for Specifying a Moving Field**

Selecting **Import** under the **B0 Input Option** in the **MHD Model** dialog box, as seen in [Figure 4.5: The MHD Model Dialog Box for Importing an External Magnetic Field \(p. 375\)](#), will result in the import of magnetic field data. The data filename can be entered in the **B0 Data File Name** field, or selected from your computer file system using the **Browse...** button. Magnetic data can also be generated using a third-party program such as MAGNA. The required format of the magnetic field data file is given in [Appendix C \(p. 389\)](#).

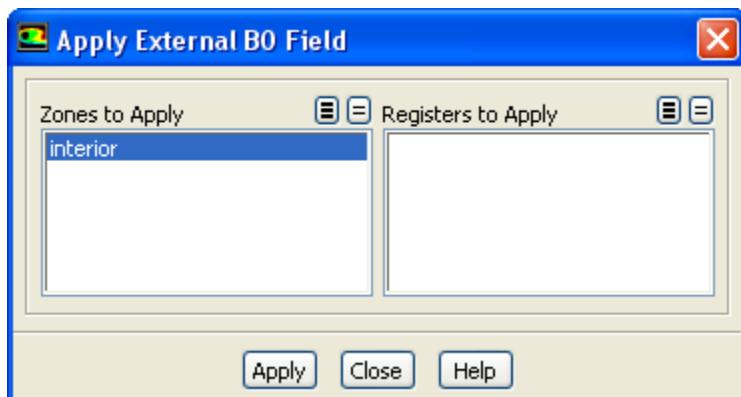
When using the **Import** option, the **B0 Data Media** is either set to **Non-Conducting** or **Conducting**, depending on the assumptions used in generating the magnetic field data. (These choices correspond to "Case 1" and "Case 2", respectively, as discussed in [Magnetic Induction Method \(p. 364\)](#).)

**Figure 4.5: The MHD Model Dialog Box for Importing an External Magnetic Field**

The **Field Type** is determined by the field data from the data file. The choice of either the **DC Field** or the **AC Field** option in the dialog box is irrelevant if the import data is either DC or AC. However, selection of both options indicates that data of both field types are to be imported from the data file, and superimposed together to provide the final external field data. Make sure that the data file contains two sections for the required data. See [Appendix C \(p. 389\)](#) for details on data file with two data sections.

The **Apply External Field...** button opens the **Apply External B0 Field** dialog box as shown in [Figure 4.6: Apply External B0 Field Dialog Box \(p. 375\)](#). To apply the external field data to zones or regions in the computational domain, select the zone names or register names of marked regions from the dialog box and click the **Apply** button.

The **Reset External Field** button sets the external magnetic field variable to zero.

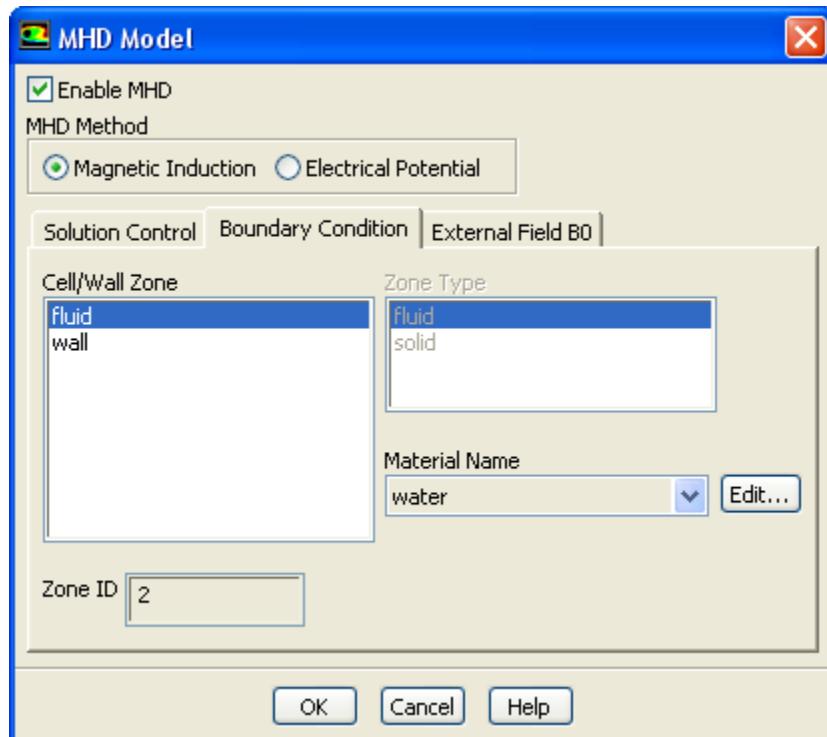
**Figure 4.6: Apply External B0 Field Dialog Box**

### 4.3.4. Setting Up Boundary Conditions

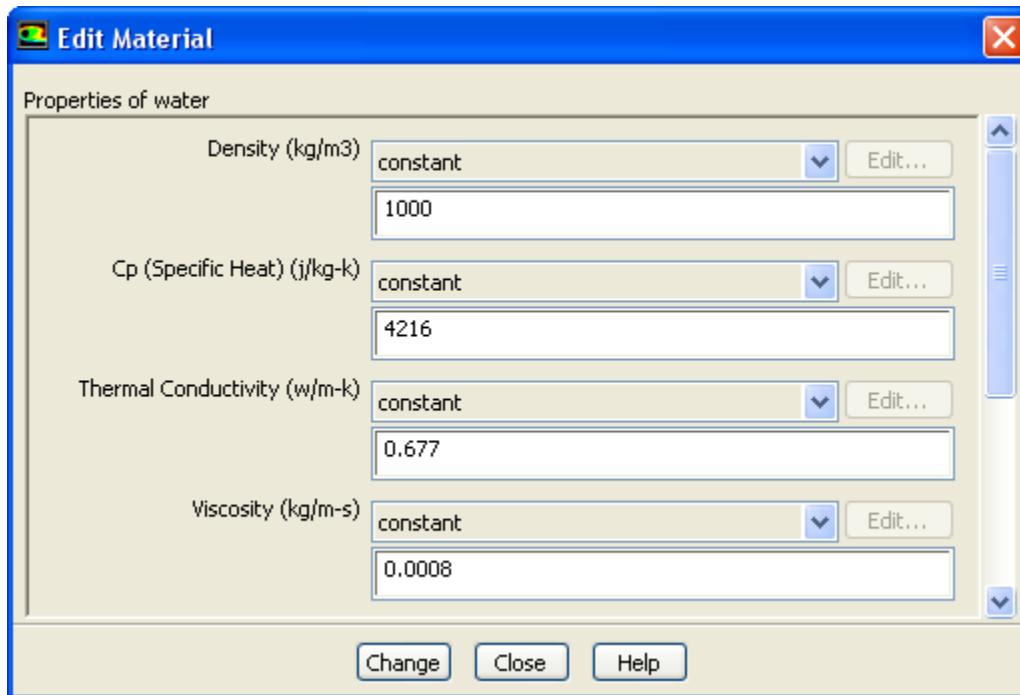
Boundary conditions related to MHD calculations are set under the **Boundary Condition** tab in the **MHD Model** dialog box. Boundary conditions can be set to cell zones and wall boundaries.

For cell zones, only the associated material can be changed and its properties modified. [Figure 4.7: Cell Boundary Condition Setup \(p. 376\)](#) shows the dialog box for cell zone boundary condition setup. The cell zone material can be selected from the **Material Name** drop-down list.

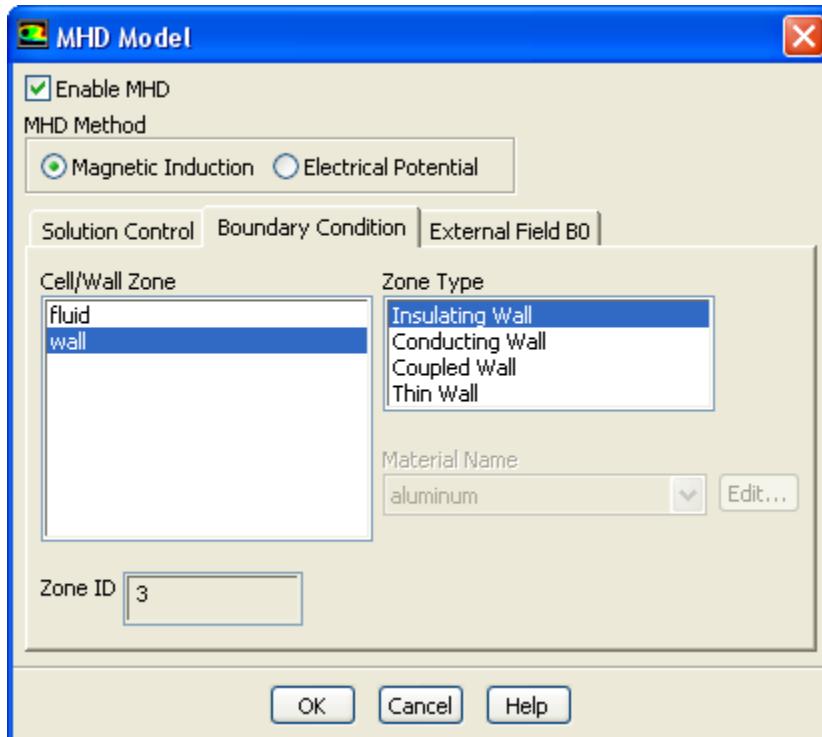
**Figure 4.7: Cell Boundary Condition Setup**



Note that the materials available in the list are set in the general ANSYS Fluent case set up. Refer to the ANSYS Fluent User Guide for details on adding materials to an ANSYS Fluent case. The properties of the selected material can be modified in the **Boundary Condition** tab by clicking **Edit...** to the right of the material name. This opens the [Edit Material dialog box](#), as shown in [Figure 4.8: Editing Material Properties within Boundary Condition Setup \(p. 377\)](#). The material properties that may be modified include the electrical conductivity and magnetic permeability. The material electrical conductivity can be set as constant, a function of temperature in forms of piecewise-linear, piecewise-polynomial or polynomial, or as a user-defined function. The material magnetic permeability can only be set as a constant.

**Figure 4.8: Editing Material Properties within Boundary Condition Setup**

For wall boundaries, the boundary condition can be set as an **Insulating Wall**, **Conducting Wall**, **Coupled Wall** or **Thin Wall**. The dialog box for wall boundary condition set up is shown in [Figure 4.9: Wall Boundary Condition Setup \(p. 377\)](#).

**Figure 4.9: Wall Boundary Condition Setup**

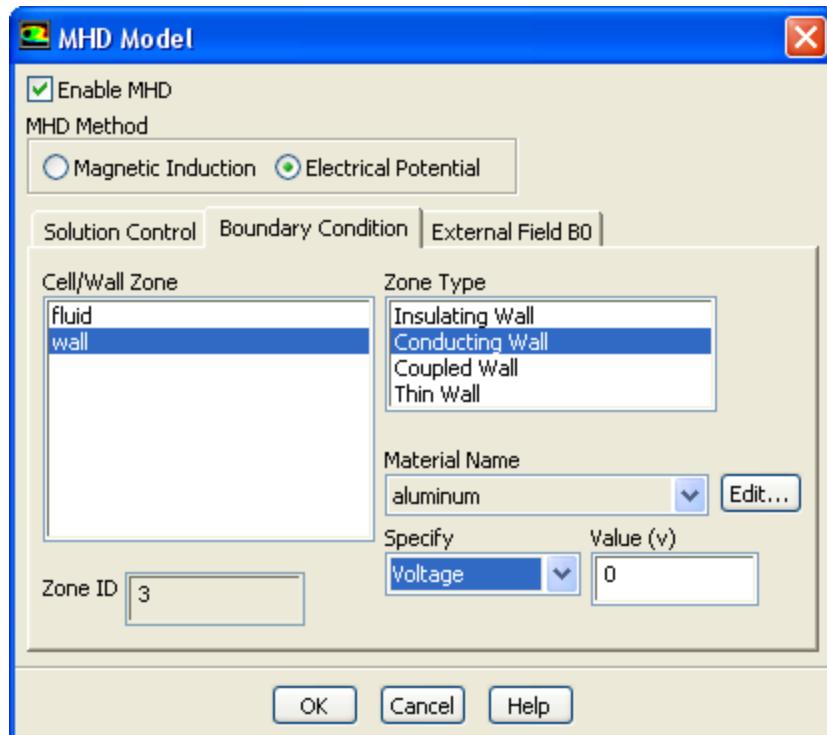
- The insulating wall is used for boundaries where there is no electric current going through the boundary.
- The conducting wall is used for boundaries that are perfect conductors.

- The coupled wall should be used for wall boundaries between solid/solid or solid/fluid zones where the MHD equations are solved.
- The thin wall type boundary can be used for external wall that has a finite electrical conductivity.

For conducting walls and thin wall boundaries, the wall material can be selected from the **Material Name** drop-down list, and its properties modified through the **Edit Material** dialog box. A wall thickness must be specified for thin wall type boundaries.

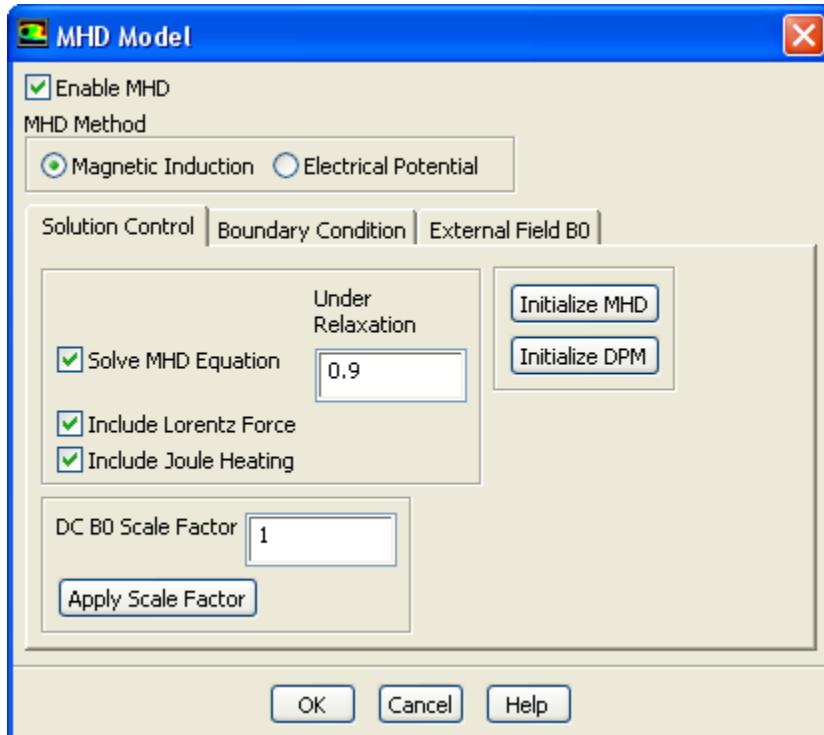
If the **Electric Potential** method is selected, the conducting wall boundary is specified by either of **Voltage** or **Current Density** at the boundary, as shown in [Figure 4.10: Conducting Wall Boundary Conditions in Electrical Potential Method \(p. 378\)](#).

**Figure 4.10: Conducting Wall Boundary Conditions in Electrical Potential Method**



### 4.3.5. Solution Controls

Under the **Solution Control** tab in the **MHD Model** dialog box, [Figure 4.11: Solution Control Tab in the MHD Model Dialog Box \(p. 379\)](#), a number of parameters can be set that control the solution process in an MHD calculation. The MHD model can be initialized using the **Initialize MHD** button. When the DPM model is enabled in the ANSYS Fluent case set up, the related variables used in the MHD model can be initialized using the **Initialize DPM** button.

**Figure 4.11: Solution Control Tab in the MHD Model Dialog Box**

You have the option to enable or disable the **Solve MHD Equation**. When the **Solve MHD Equation** is enabled, you have the choice to **Include Lorentz Force** and/or **Include Joule Heating** in the solution of flow momentum and energy equations. The under-relaxation factor for the MHD equations can also be set.

The strength of the imposed external magnetic field can be adjusted by specifying and applying scale factors to the external DC and/or AC magnetic field data.

## 4.4. MHD Solution and Postprocessing

The following sections describe the solution and postprocessing steps for the MHD model:

- 4.4.1. MHD Model Initialization
- 4.4.2. Iteration
- 4.4.3. Postprocessing

### 4.4.1. MHD Model Initialization

Initialization of the MHD model involves setting the externally-imposed magnetic field and initializing all MHD related user-defined scalars and memory variables.

When an ANSYS Fluent case is initialized, all user-defined scalar and memory variables are set to zero. The external magnetic field data is set from the **External Field B0** tab in the **MHD Model** dialog box. The **Initialize MHD** button under the **Solution Control** tab can be used to initialize the model during an ANSYS Fluent solution process. It is used when MHD effects are added to a fully or partially solved flow field, or when the model parameters are changed during an MHD calculation. It only clears the scalar variables and most of the memory locations used in the MHD model, the memory variables for the external magnetic field data are preserved.

## 4.4.2. Iteration

It is often an effective strategy to begin your MHD calculations using a previously-converged flow field solution. With this approach, the induction equations themselves are generally easy to converge. The under-relaxation factors for these equations can be set to  $0.8 \sim 0.9$ , although for very strong magnetic fields, smaller values may be needed. For the electric potential equation, the convergence is generally slow. However, the under-relaxation value for this equation should not be set to 1. As additional source terms are added to the momentum and energy equations, the under-relaxation factors for these equations should generally be reduced to improve the rate of convergence. In case of convergence difficulties, another helpful strategy is to use the **B0 Scale Factor** in the **Solution Control** tab ([Figure 4.2: The MHD Model Dialog Box \(p. 371\)](#)). This will gradually increase the MHD effect to its actual magnitude through a series of restarts. When the strength of the externally imposed magnetic field is strong, it is advisable to start the calculation with a reduced strength external field by applying a small scale factor. When the calculation is approaching convergence the scale factor can be increased gradually until the required external field strength is reached.

## 4.4.3. Postprocessing

You can use the standard postprocessing facilities of ANSYS Fluent to display the MHD calculation results.

Contours of MHD variables can be displayed using

**Results** → **Graphics** → **Contours** **Edit...**

The MHD variables can be selected from the variable list.

Vectors of MHD variables, such as the magnetic field vector and current density vector, can be displayed using

**Results** → **Graphics** → **Vectors** **Edit...**

The vector fields of the MHD variables are listed in the **Vectors of** drop-down list in the **Vectors** dialog box. [Table 4.2: MHD Vectors \(p. 380\)](#) lists the MHD related vector fields.

**Table 4.2: MHD Vectors**

Name	Unit	Description
Induced- $\vec{B}$ -Field	Tesla	Induced magnetic field vector
External- $\vec{B}$ -Field	Tesla	Applied external magnetic field vector
Current-Density $\vec{J}$	A/m <sup>2</sup>	Induced current density field vector
Electric-Field $\vec{E}$	V/m	Electric field density vector
Lorentz-Force $\vec{F}$	N/m <sup>3</sup>	Lorentz force vector

### Note

- For field variables that are stored in UDM, use the corresponding variables for post processing. Post processing the UDM itself is not recommended.

- For reviewing simulation results in CFD Post for cases involving UDM or UDS, export the solution data as CFD-Post-compatible file (.cdat) in Fluent and then load this file into CFD-Post.

## 4.5. Limitations

Many MHD applications involve the simultaneous use of other advanced ANSYS Fluent capabilities such as solidification, free surface modeling with the volume of fluid (VOF) approach, DPM, Eulerian multiphase, and so on. You should consult the latest ANSYS Fluent documentation for the limitations that apply to those features. In addition, you should be aware of the following limitations of the MHD capability.

- As explained in [Magnetohydrodynamic Model Theory \(p. 363\)](#), the MHD module assumes a sufficiently conductive fluid so that the charge density and displacement current terms in Maxwell's equations can be neglected. For marginally conductive fluids, this assumption may not be valid. More information about this simplification is available in the bibliography.
- For electromagnetic material properties, only constant isotropic models are available. Multiphase volume fractions are not dependent on temperature, species concentration, or field strength. However, sufficiently strong magnetic fields can cause the constant-permeability assumption to become invalid.
- You must specify the applied magnetic field directly. The alternative specification of an imposed electrical current is not supported.
- In the case of alternating-current (AC) magnetic fields, the capability has been designed for relatively low frequencies; explicit temporal resolution of each cycle is required. Although not a fundamental limitation, the computational expense of simulating high-frequency effects may become excessive due to small required time step size. Time-averaging methods to incorporate high-frequency MHD effects have not been implemented.



---

---

## Appendix A. Guidelines For Using the ANSYS Fluent MHD Model

This appendix provides a basic outline for installing the magnetohydrodynamics (MHD) module and solving MHD problems in ANSYS Fluent.

### A.1. Installing the MHD Module

#### A.2. An Overview of Using the MHD Module

---

##### Important

While [Using the ANSYS Fluent MHD Module \(p. 369\)](#) covers much of the same material in greater detail, this appendix presents a set of guidelines for solving typical MHD problems with ANSYS Fluent, with occasional references to [Using the ANSYS Fluent MHD Module \(p. 369\)](#) where more information can be found.

---

## A.1. Installing the MHD Module

Before using the MHD module, you first need to install the necessary files onto your computer. These files are provided with your standard installation of ANSYS Fluent. They can be found in your installation area in a directory called addons/mhd. The MHD module is loaded into ANSYS Fluent through the text user interface (TUI)

```
define → models → addon-module
```

only after a valid ANSYS Fluent case file has been set or read.

Once the MHD model is installed, beneath the mhd directory there are two subdirectories: a lib directory, and a directory corresponding to your specific architecture, ntx86 for example. The lib directory holds a Scheme code called addon.bin that contains the MHD module graphical interface. The specific architecture directory, ntx86 for example, contains the following subdirectories that hold various ANSYS Fluent files:

```
2d      2ddp     3d      3ddp  
2d_host 2ddp_host 3d_host 3ddp_host  
2d_node 2ddp_node 3d_node 3ddp_node
```

## A.2. An Overview of Using the MHD Module

To use the MHD module in an ANSYS Fluent simulation, follow the general guidelines:

1. Start ANSYS Fluent.

To begin modeling your MHD simulation, you need to start an appropriate ANSYS Fluent session. Choose from either the 2D, 3D, **Double Precision**, or the parallel version of ANSYS Fluent.

2. Read in a mesh file or a case file.

You can have ANSYS Fluent read in your mesh file, a previously saved non-MHD case file, or a previously saved MHD case file.

---

### Important

Note that if you read in a new mesh file, you need to perform the appropriate mesh check and mesh scale procedures.

---

### 3. Load the MHD module.

The MHD module is loaded into ANSYS Fluent using the text command

define → models → addon-module

and entering the corresponding module number ([Loading the MHD Module \(p. 369\)](#)).

### 4. Set up the MHD model.

The **MHD Model** dialog box is accessed using the graphical user interface (GUI):

**Setup** → **Models** → **MHD Model** **Edit...**

If the MHD model is not enabled after the MHD module is loaded for the first time, you can enable it by clicking the **Enable MHD** button, which will display the expanded dialog box ([Enabling the MHD Model \(p. 371\)](#)).

### 5. Select an MHD method.

The method used for the MHD calculation can be selected under **MHD Method**. The two methods are

- **Magnetic Induction** ([Magnetic Induction Method \(p. 364\)](#))
- **Electrical Potential** ([Electric Potential Method \(p. 365\)](#))

### 6. Apply an external magnetic field.

This is done by entering values for the B0 components in the **External Field B0** tab. B0 input options can either be

- **Patched**, or
- **Imported**

The **Field Type** will either be the **DC Field** or the **AC Field**. The **Field Type** is determined by the field data from the data file. Refer to [Applying an External Magnetic Field \(p. 372\)](#) for details on applying an external magnetic field.

### 7. Set up the boundary conditions.

Under the **Boundary Condition** tab, cell zones and wall boundaries can be selected as well as the corresponding zone type.

Cell zone materials are selected from the **Material Name** drop-down list. The properties of the selected material can be modified by clicking on the **Edit...** button to the right of the material name. Note that the materials available in the list are set in the general ANSYS Fluent case setup.

### **Setup** → **Materials**

The material properties that may be modified include the electrical conductivity and magnetic permeability.

Wall boundary conditions can be set as an **Insulating Wall**, **Conducting Wall**, **Coupled Wall** or **Thin Wall** (see [Setting Up Boundary Conditions \(p. 376\)](#)).

## 8. Set solution controls.

Under the **Solution Control** tab:

- The MHD equation is enabled or disabled.
- Lorentz force and Joule heat sources are enabled or disabled.
- under-relaxation factors are set (reasonable under-relaxation factors for the MHD equations are  $0.8 \sim 0.9$ ).
- Scale factors can be used to adjust the strength of the imposed external magnetic field. As the calculation approaches convergence, the scale factor in the **Solution Control** tab can be increased gradually until the required external field strength is reached ([Solution Controls \(p. 378\)](#)).
- The MHD model is initialized ([MHD Model Initialization \(p. 379\)](#)).

## 9. Run the ANSYS Fluent MHD simulation.

### **Solution** → **Run Calculation**

Set the number of iterations. It is often an effective strategy to begin your MHD calculations using a previously-converged flow field solution. With this approach, the induction equations themselves are generally easy to converge. For more information, see [Iteration \(p. 380\)](#).

## 10. Process the solution data.

You can use the standard postprocessing facilities of ANSYS Fluent to display the results of an MHD calculation. Contours of MHD variables can be displayed.

### **Results** → **Graphics** → **Contours** **Edit...**

The MHD variables can be selected from the variable list. Vectors of MHD variables, such as the magnetic field vector and current density vector, can be displayed using custom vectors.

### **Results** → **Graphics** → **Vectors** **Edit...**

For more information, see [Postprocessing \(p. 380\)](#).



---

---

## Appendix B. Definitions of the Magnetic Field

The sinusoidal form of the magnetic field is defined as:

$$\begin{aligned}B_0 &= \bar{B}_0 + A_0 \cos(2\pi ft - K \cdot R + \phi) \\K &= \frac{1}{\lambda} \left\{ \frac{1}{\cos\alpha} i + \frac{1}{\cos\beta} j + \frac{1}{\cos\gamma} k \right\}\end{aligned}\quad (1)$$

where  $\bar{B}_0$  is the mean vector,  $A_0$  is the amplitude vector,  $K$  is defined as the propagation vector,  $R$  is the position vector of an arbitrary point.  $\cos\alpha$ ,  $\cos\beta$  and  $\cos\gamma$  are the  $x$ ,  $y$  and  $z$  direction cosines respectively. The quantities  $f$ ,  $\lambda$ , and  $\phi$  are the frequency, wavelength, and phase offset, respectively. For a non-moving field the propagation vector is zero. For a static field only applies.

The square form of the magnetic field is defined as:

$$B_0 = \bar{B}_0 + A_0 \frac{\cos(2\pi ft - K \cdot R + \phi)}{|\cos(2\pi ft - K \cdot R + \phi)|} \quad (2)$$

The definition of the propagation vector is the same as for the sinusoidal form.



---

---

## Appendix C. External Magnetic Field Data Format

The external magnetic field data file is in text format and of the following structure:

*MAG-DATA*

```
nX      nY      nZ  
X1      Xn  
Y1      Yn  
Z1      Zn  
nAC     Freq  
BXre-1  BYre-1  BZre-1  BXim-1  BYim-1  BZim-1  
...  
BXre-n  BYre-n  BZre-n  BXim-n  BYim-n  BZim-n
```

The first line is an identification tag for the data file. The second line defines the number of data points in the *x*, *y* and *z* directions. The next three lines define the ranges in *x*, *y* and *z* directions. The data points are assumed to be evenly distributed along each direction. Line 6 defines the AC field flag and frequency. When *nAC* = 0, the magnetic field is static. For AC field, *nAC* = 1 and *Freq* is the frequency in Hz.

The rest of the data file contains the magnetic field data points. Each line defines the components of the real and imaginary parts of the magnetic field vector on one data point. The data points are indexed as:

$$\begin{aligned} l &= i + nX((j-1) + nY(k-1)) \\ i &= 1, \dots, nX; j = 1, \dots, nY; k = 1, \dots, nZ \end{aligned} \tag{1}$$

The data is listed in the ascending order from 1 to *n*, where *n* is the total number of data points given by *n* = *nXnYnZ*.

For magnetic fields composed of both DC and AC fields, the entire file structure described above is repeated for the DC and AC parts. These two sections within the same file will be imported into ANSYS Fluent and stored separately. The order of the DC and AC sections of the file is not important.

The imported data is interpreted as a snapshot of the applied magnetic field at an instant in time. Complex form is used to accommodate oscillating/moving fields. Thus, using complex numbers, and with reference to the quantities defined in [Appendix B \(p. 387\)](#),

$$\begin{aligned} \vec{B}_0 &\equiv \begin{Bmatrix} BX_{re} \\ BY_{re} \\ BZ_{re} \end{Bmatrix} + i \begin{Bmatrix} BX_{im} \\ BY_{im} \\ BZ_{im} \end{Bmatrix} \\ &= \bar{B}_0 + A_0 \exp[i(2\pi f t + \phi)] \end{aligned} \tag{2}$$

For a DC field,

$$\bar{B}_{o,i} = B_{re,i} \quad i=x,y,z \quad (3)$$

For an AC field,

$$A_{o,i} = \sqrt{B_{re,i}^2 + B_{im,i}^2} \quad i=x,y,z \quad (4)$$

and

$$\phi_i = t g^{-1} \left( \frac{B_{im,i}}{B_{re,i}} \right) \quad i=x,y,z \quad (5)$$

Note that when the external magnetic field import option is used, the frequency,  $f$ , read from this file supersedes the value specified in the GUI.

---

---

## Appendix D. MHD Module Text Commands

**mhd-models/**

Define solver configuration.

**enable-mhd?**

Enable/disable MHD model.

**mhd-method**

Select MHD method.

**boundary-conditions/**

Define MHD boundary conditions

**list-zones**

List ANSYS Fluent zone information.

**fluid**

Set fluid zone boundary condition.

**solid**

Set solid zone boundary condition.

**wall**

Set wall boundary condition.

**b0-scale-factor**

Set and apply external magnetic field scale factor.

**external-b0-field**

Set and apply external magnetic field data.

**initialize-mhd**

Initialize MHD model.

**initialize-dpm**

Initialize DPM related MHD variables.

**solution-control**

Set MHD solution control parameters.



---

# Bibliography

- [1] R. Moreau. *Magnetohydrodynamics*. Kluwer Academic Publishers. 1990.



---

---

## **Part VII: ANSYS Fluent Population Balance Module**

[Using This Manual \(p. cccxcvii\)](#)

[1. Introduction \(p. 399\)](#)

[2. Population Balance Model Theory \(p. 403\)](#)

[3. Using the ANSYS Fluent Population Balance Model \(p. 423\)](#)

[4. Postprocessing for the Population Balance Model \(p. 443\)](#)

[5. UDFs for Population Balance Modeling \(p. 447\)](#)

[Appendix A.DEFINE\\_HET\\_RXN\\_RATE Macro \(p. 455\)](#)

[Bibliography \(p. 459\)](#)

---

---

---

# Using This Manual

---

## 1. The Contents of This Manual

The ANSYS Fluent Population Balance Model Manual tells you what you need to know to model population balance with ANSYS Fluent. In this manual you will find background information pertaining to the model, a theoretical discussion of the model used in ANSYS Fluent, and a description of using the model for your CFD simulations.

This part is divided into the following chapters:

- [Introduction \(p. 399\)](#)
- [Population Balance Model Theory \(p. 403\)](#)
- [Using the ANSYS Fluent Population Balance Model \(p. 423\)](#)
- [Postprocessing for the Population Balance Model \(p. 443\)](#)
- [UDFs for Population Balance Modeling \(p. 447\)](#)
- [Appendix A \(p. 455\)](#)
- [Bibliography \(p. 459\)](#)



---

## Chapter 1: Introduction

---

In ANSYS Fluent the population balance model is provided as an add-on module with the standard ANSYS Fluent licensed software.

Several industrial fluid flow applications involve a secondary phase with a size distribution. The size distribution of particles, including solid particles, bubbles, or droplets, can evolve in conjunction with transport and chemical reaction in a multiphase system. The evolutionary processes can be a combination of different phenomena like nucleation, growth, dispersion, dissolution, aggregation, and breakage producing the dispersion. Thus in multiphase flows involving a size distribution, a balance equation is required to describe the changes in the particle population, in addition to momentum, mass, and energy balances. This balance is generally referred to as the population balance. Cases in which a population balance could apply include crystallization, precipitative reactions from a gas or liquid phase, bubble columns, gas sparging, sprays, fluidized bed polymerization, granulation, liquid-liquid emulsion and separation, and aerosol flows.

To make use of this modeling concept, a number density function is introduced to account for the particle population. With the aid of particle properties (for example, particle size, porosity, composition, and so on), different particles in the population can be distinguished and their behavior can be described.

ANSYS Fluent offers three solution methods to the population balance equation: discretized population balance, standard method of moments, and quadrature method of moments.

- 1.1.The Discrete Method
- 1.2.The Inhomogeneous Discrete Method
- 1.3.The Standard Method of Moments
- 1.4.The Quadrature Method of Moments

### 1.1. The Discrete Method

In the discrete method, the particle population is discretized into a finite number of size intervals. This approach has the advantage of computing the particle size distribution (PSD) directly. This approach is also particularly useful when the range of particle sizes is known *a priori* and does not span more than two or three orders of magnitude. In this case, the population can be discretized with a relatively small number of size intervals and the size distribution that is coupled with fluid dynamics can be computed. The disadvantage of the discrete method is that it is computationally expensive if a large number of intervals is needed.

### 1.2. The Inhomogeneous Discrete Method

One of the limitations of the existing homogeneous discrete method is that all bins are assigned to the same secondary phase and are therefore advected with the same phase momentum. This is unsuitable for modeling cases where large and small bin sizes are likely to segregate due to different momentum fields. The inhomogeneous discrete method overcomes this limitation by allowing groups of bins to be advected by different phase velocities. Therefore when the inhomogeneous discrete model is activated, the Population Balance model can be applied to more than one secondary phase.

The general transport equation for the discrete bin fraction  $f_i$  can be written as

$$\frac{\partial}{\partial t} (\rho \alpha f_i) + \nabla \cdot (\vec{u}_p \alpha f_i) = S_{bi} \quad (1.1)$$

Since all bins belong to a single phase in the homogeneous discrete method, the net mass source for the phase in case of breakage and agglomeration is zero and can be expressed as

$$\sum_{i=1}^M S_{bi} = 0 \quad (1.2)$$

This is shown schematically in [Figure 1.1: Homogeneous Discrete Method \(p. 401\)](#) where all bins are advected by the same phase velocity  $u_p$ . In contrast, the inhomogeneous discrete method shown in [Figure 1.2: Inhomogeneous Discrete Method \(p. 401\)](#) allows bins to be assigned to multiple phases. Here  $M$  bins per phase are distributed over  $N$  phases for a total of  $M \times N$  bins. Bins  $f_1$  and  $f_M$  are advected by phase velocity  $u_{p_1}$  and so forth. The sum of bin sources in any given phase is not necessarily equal to zero since bins in a given phase can migrate to another phase through breakage or agglomeration, thereby creating a net mass source for that phase.

The net mass source for a given phase can be expressed as the sum of the bin sources belonging to that phase

$$S_i = \sum_{i=1}^M S_{bi} \quad (1.3)$$

For breakage and coalescence the sum over all phase sources is zero

$$\sum_{i=1}^N S_i = 0 \quad (1.4)$$

also similar to the homogeneous discrete model

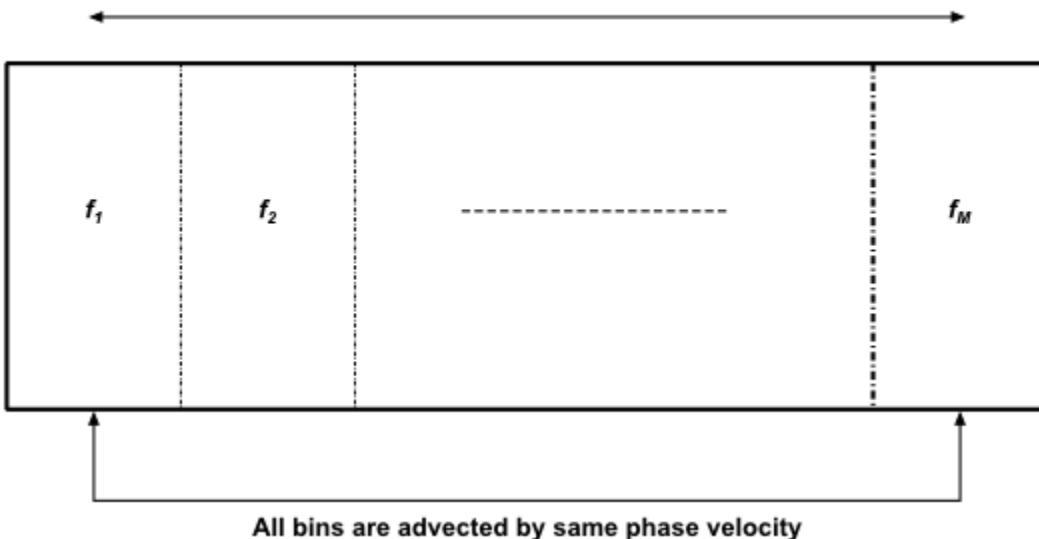
$$\sum_{i=1}^M f_i = 1 \quad (1.5)$$

### Important

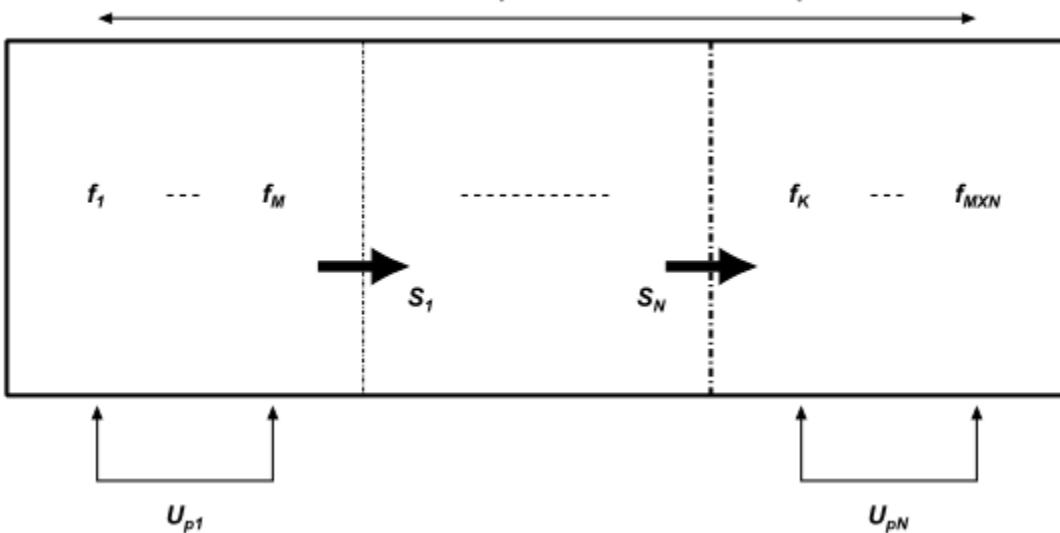
The inhomogeneous discrete method is currently limited to breakage and coalescence only.

**Figure 1.1: Homogeneous Discrete Method**

Bins interact with one another through breakage, agglomeration

**Figure 1.2: Inhomogeneous Discrete Method**

Bins interact within each phase as well as across phases



### 1.3. The Standard Method of Moments

The standard method of moments (SMM) is an efficient alternative to the discrete population balance approach. In this approach, the population balance equation is transformed into a set of transport equations for moments of the distribution. The  $i^{\text{th}}$  moment is defined by integrating the number density throughout the particle space weighted with the particle property raised to its  $i^{\text{th}}$  power. It is generally sufficient to solve only a few moment equations, typically three to six. This may provide a significant reduction in the number of equations to be solved compared with the discretized approach. Apart from the computational advantage, the SMM approach is useful when the entire distribution is not needed and certain average and total quantities are sufficient to represent the particle distribution. Typically, the zeroth moment represents the total number density, the second moment represents the total surface area per unit volume, and the third moment represents the total mass density.

In the SMM approach, no assumptions are made about the size distribution, and the moment equations are formulated in a closed form involving only functions of the moments themselves. However, this exact closure requirement poses a serious limitation, as aggregation (with the exception of the constant aggregation kernel) and breakage phenomena cannot be written as functions of moments.

## 1.4. The Quadrature Method of Moments

The quadrature method of moments (QMOM) has a similar advantage as the SMM in terms of computational costs, but replaces the exact closure needed by SMM with an approximate closure. This allows application of QMOM to a broad range of applications without any limitations.

---

## Chapter 2: Population Balance Model Theory

---

This chapter presents an overview of the theory and the governing equations for the methods used in ANSYS Fluent to predict particle growth and nucleation.

- 2.1. The Particle State Vector
- 2.2. The Population Balance Equation (PBE)
- 2.3. Solution Methods
- 2.4. Population Balance Statistics

### 2.1. The Particle State Vector

The particle state vector is characterized by a set of "external coordinates" ( $\vec{x}$ ), which denote the spatial position of the particle, and "internal coordinates" ( $\phi$ ), which could include particle size, composition, and temperature. From these coordinates, a number density function  $n(\vec{x}, \phi, t)$  can be postulated where  $\phi \in \Omega_\phi$ ,  $\vec{x} \in \Omega_{\vec{x}}$ . Therefore, the average number of particles in the infinitesimal volume  $dV_{\vec{x}} dV_\phi$  is  $n(\vec{x}, \phi, t) dV_{\vec{x}} dV_\phi$ . In contrast, the continuous phase state vector is given by

$$\vec{Y} \equiv [Y_1(\vec{x}, t), Y_2(\vec{x}, t), \dots, Y_c(\vec{x}, t)]$$

The total number of particles in the entire system is then

$$\int_{\Omega_\phi} \int_{\Omega_{\vec{x}}} n dV_{\vec{x}} dV_\phi \quad (2.1)$$

The local average number density in physical space (that is, the total number of particles per unit volume) is given by

$$N(\vec{x}, t) = \int_{\Omega_\phi} n dV_\phi \quad (2.2)$$

The total volume fraction of all particles is given by

$$\alpha(\vec{x}, t) = \int_{\Omega_\phi} n V(\phi) dV_\phi \quad (2.3)$$

where  $V(\phi)$  is the volume of a particle in state  $\phi$ .

### 2.2. The Population Balance Equation (PBE)

Assuming that  $\phi$  is the particle volume, the transport equation for the number density function is given as

$$\begin{aligned}
 & \frac{\partial}{\partial t}[n(V,t)] + \nabla \cdot [\vec{u}n(V,t)] + \underbrace{\nabla_v \cdot [G_v n(V,t)]}_{\text{Growth term}} \\
 &= \underbrace{\frac{1}{2} \int_0^V a(V-V', V') n(V-V', t) n(V', t) dV'}_{\text{Birth due to Aggregation}} \\
 &\quad - \underbrace{\int_0^\infty a(V, V') n(V, t) n(V', t) dV'}_{\text{Death due to Aggregation}} \\
 &\quad + \underbrace{\int_{\Omega_v} p g(V') \beta(V|V') n(V', t) dV'}_{\text{Birth due to Breakage}} \\
 &\quad - \underbrace{g(V) n(V, t)}_{\text{Death due to Breakage}}
 \end{aligned} \tag{2.4}$$

The boundary and initial conditions are given by

$$n(V, t=0) = n_v; n(V=0, t) G_v = \dot{n}_0 \tag{2.5}$$

where  $\dot{n}_0$  is the nucleation rate in particles/m<sup>3</sup>-s.

### 2.2.1. Particle Growth and Dissolution

The growth rate based on particle volume,  $G_v$ , (m<sup>3</sup>/s) is defined as

$$G_v = \frac{\partial V}{\partial t} \tag{2.6}$$

The growth rate based on particle diameter (or length) is defined as

$$G = \frac{\partial L}{\partial t} \tag{2.7}$$

The volume of a single particle  $V$  is defined as  $K_v L^3$ , and therefore the relationship between  $G_v$  and  $G$  is

$$G_v = 3K_v L^2 G \tag{2.8}$$

The surface area of a single particle,  $A$ , is defined as  $K_a L^2$ . Thus for a cube or a sphere,  $K_a = 6K_v$ .

#### Important

Dissolution of particles can be represented as negative growth.

### 2.2.2. Particle Birth and Death Due to Breakage and Aggregation

The birth and death of particles occur due to breakage and aggregation processes. Examples of breakage processes include crystal fracture in crystallizers and bubble breakage due to liquid turbulence in a bubble column. Similarly, aggregation can occur due to particle agglomeration in crystallizers and bubble coalescence in bubble column reactors.

### 2.2.2.1. Breakage

The breakage rate expression, or kernel [19] (p. 460), is expressed as

$$g(V')\beta(V|V')$$

where

$g(V')$  = breakage frequency; that is, the fraction of particles of volume  $V'$  breaking per unit time ( $s^{-1}$ )

$\beta(V|V')$  = probability density function (PDF) of particles breaking from volume  $V'$  to a particle of volume  $V$

The birth rate of particles of volume  $V$  due to breakage is given by

$$B_{br} = \int_{\Omega_v} pg(V')\beta(V|V')n(V')dV' \quad (2.9)$$

where  $g(V')n(V')dV'$  particles of volume  $V'$  break per unit time, producing  $pg(V')n(V')dV'$  particles, of which a fraction  $\beta(V|V')dV$  represents particles of volume  $V$ .  $p$  is the number of child particles produced per parent (for example, two particles for binary breakage).

The death rate of particles of volume  $V$  due to breakage is given by

$$D_{br} = g(V)n(V) \quad (2.10)$$

The PDF  $\beta(V|V')$  is also known as the particle fragmentation distribution function, or daughter size distribution. Several functional forms of the fragmentation distribution function have been proposed, though the following physical constraints must be met: the normalized number of breaking particles must sum to unity, the masses of the fragments must sum to the original particle mass, and the number of fragments formed has to be correctly represented.

Mathematically, these constraints can be written as follows:

- For the normalization condition:

$$\int_0^{V'} \beta(V|V')dV = 1 \quad (2.11)$$

- For conservation of mass

$$p \int_0^{V'} m(V) \beta(V|V')dV = m(V') \quad (2.12)$$

- For binary breakage,  $\beta$  is symmetric about  $V/V' = 0.5$ ; that is,

$$\beta(V'-V|V') = \beta(V|V') \quad (2.13)$$

The following is a list of models available in ANSYS Fluent to calculate the breakage frequency:

- Constant value

- Luo model
- Lehr model
- Ghadiri model
- Laakkonen model
- User-defined model

ANSYS Fluent provides the following models for calculating the breakage PDF:

- Parabolic PDF
- Laakkonen PDF
- Generalized PDF for multiple breakage fragments
- User-defined model

The breakage frequency models and the parabolic and generalized PDFs are described in detail in the sections that follow.

### **2.2.2.2. Luo and Lehr Breakage Kernels**

The Luo and Lehr models are integrated kernels, encompassing both the breakage frequency and the PDF of breaking particles. The general breakage rate per unit volume is usually written [16] (p. 459) as

$$\Omega_{br}(V, V') = \Omega_B(V') \eta(V|V') [1/m^3/sec] \quad (2.14)$$

where the original particle has a volume  $V'$  and the daughter particle has a volume  $V$ . In the previous expression,  $\Omega_B(V')$  is the breakage frequency, and  $\eta(V|V')$  is the normalized daughter particle distribution function. For binary breakage, the breakage kernel must be symmetrical with respect to  $\frac{V}{V'} = 0.5$ .

The general form is the integral over the size of eddies  $\lambda$  hitting the particle with diameter  $d$  (and volume  $V$ ). The integral is taken over the dimensionless eddy size  $\xi = \lambda/d$ . The general form is

$$\Omega_{br}(V, V') = K \int_{\xi_{min}}^1 \frac{(1+\xi)^2}{\xi^n} \exp(-b\xi^m) d\xi \quad (2.15)$$

where the parameters are as shown in the tables that follow:

**Table 2.1: Luo Model Parameters**

$K[1/m^3/sec]$	$n$	$b$	$m$
$0.9238\varepsilon^{1/3}d^{-2/3}\alpha$	$11/3$	$12[f^{2/3} + (1-f)^{2/3} - 1]\sigma\rho^{-1}\varepsilon^{-2/3}d^{-5/3}\beta^{-1}$	$-11/3$

Where  $\beta=2.047$ . See Luo [19] (p. 460).

**Table 2.2: Lehr Model Parameters**

$K[1/m^3/sec]$	$n$	$b$	$m$
$1.19\varepsilon^{-1/3}d^{7/3}\sigma\rho^{-1}f^{-1/3}$	$13/3$	$2We_{crit}\sigma\rho^{-1}\varepsilon^{-2/3}d^{-5/3}f^{-1/3}$	$-2/3$

Where  $We_{crit}$  is entered through the GUI. See Lehr [16] (p. 459).

### 2.2.2.3. Ghadiri Breakage Kernels

The Ghadiri model [8] (p. 459), [23] (p. 460), in contrast to the Luo and Lehr models, is used to model only the breakage frequency of the solid particles. You will have to specify the PDF model to define the daughter distribution.

The breakage frequency  $f$  is related to the material properties and impact conditions:

$$f = \frac{\rho_s E^{2/3}}{\Gamma^{5/3}} v^2 L^{5/3} = K_b v^2 L^{5/3} \quad (2.16)$$

where  $\rho_s$  is the particle density,  $E$  is the elastic modulus of the granule, and  $\Gamma$  is the interface energy.  $v$  is the impact velocity and  $L$  is the particle diameter prior to breaking.  $K_b$  is the breakage constant and is defined as

$$K_b = \frac{\rho_s E^2 / 3}{\Gamma^{5/3}} \quad (2.17)$$

### 2.2.2.4. Laakkonen Breakage Kernels

The Laakkonen breakage kernel is expressed as the product of a breakage frequency,  $g(V')$  and a daughter PDF  $\beta(V, V')$  where

$$g(V') = C_2 \varepsilon^{1/3} \operatorname{erfc} \left( \sqrt{C_3 \frac{\sigma}{\rho_L \varepsilon^{2/3} d^{5/3}} + C_4 \frac{\mu_L}{\sqrt{\rho_L \rho_g} \varepsilon^{2/3} d^{5/3}}} \right) \quad (2.18)$$

where  $\varepsilon$  is the liquid phase eddy dissipation,  $\sigma$  is the surface tension,  $\rho_L$  is the liquid density,  $\rho_g$  is the gas density,  $d$  is the parent particle diameter and  $\mu_L$  is the liquid viscosity.

The constants  $C_2=2.52$ ,  $C_3=0.04$  and  $C_4=0.01$ .

The daughter PDF is given by

$$\beta(V, V') = \frac{30}{V'} \left( \frac{V}{V'} \right)^2 \left( 1 - \frac{V}{V'} \right)^2 \quad (2.19)$$

where  $V$  and  $V'$  are the daughter and parent particle volumes, respectively. This model is a useful alternative to the widely used Luo model because it has a simple expression for the daughter PDF and therefore requires significantly less computational effort.

### 2.2.2.5. Parabolic PDF

The breakage PDF function contains information on the probability of fragments formed by a breakage event. It provides the number of particles and the possible size distribution from the breakage. The parabolic form of the PDF implemented in ANSYS Fluent enables you to define the breakage PDF such that

$$\beta(V|V') = 0.5 \left[ \frac{C}{V'} + \frac{1-C/2}{V'} \left\{ 24 \left( \frac{V}{V'} \right)^2 - 24 \left( \frac{V}{V'} \right) + 6 \right\} \right] \quad (2.20)$$

where  $V$  and  $V'$  are the daughter and parent particle volumes, respectively. Depending on the value of the shape factor  $C$ , different behaviors will be observed in the shape of the particle breakage distribution function. For example, if  $C=2$ , the particle breakage has a uniform distribution. If  $0 < C < 2$ , a concave parabola is obtained, meaning that it is more likely to obtain unequally-sized fragments than equally-sized fragments. The opposite of this is true for  $2 < C < 3$ . Values outside of the range of 0 to 3 are not allowed, because the PDF cannot have a negative value.

Note that the PDF defined in [Equation 2.20 \(p. 408\)](#) is symmetric about  $V/V' = 0.5$ .

### 2.2.2.6. Generalized PDF

The generalized form of the PDF implemented in ANSYS Fluent enables you to simulate multiple breakage fragments ( $>2$ ) and to specify the form of the daughter distribution (for example, uniform, equal-size, attrition, power law, parabolic, binary beta). The model itself can be applied to both the discrete method and the QMOM.

Considering the self-similar formulation [\[26\] \(p. 460\)](#) where the similarity  $z$  is the ratio of daughter-to-parent size (that is,  $z \equiv \frac{V}{V'}$ ), then the generalized PDF is given by

$$p\beta(V|V') = \frac{\theta(z)}{V'} \quad (2.21)$$

where  $\theta(z)$  is the self-similar daughter distribution [\[6\] \(p. 459\)](#). The  $k^{th}$  moment of  $\theta(z)$ ,  $b_k$ , is

$$b_k = \int_0^1 z^k \theta(z) dz = \frac{B_k(V')}{V'^k} \quad (2.22)$$

where

$$B_k(V') = \int_0^{V'} V^k p\beta(V|V') dV \quad (2.23)$$

The conditions of number and mass conservation can then be expressed as

$$b_0 = \int_0^1 \theta(z) dz = p \quad (2.24)$$

$$b_1 = \int_0^1 z \theta(z) dz = 1 \quad (2.25)$$

The generalized form of  $\theta(z)$  [\[6\] \(p. 459\)](#) can be expressed as

$$\theta(z) = \sum_i w_i p_i \frac{z^{q_i-1} (1-z)^{r_i-1}}{\beta(q_i, r_i)} \quad (2.26)$$

where  $i$  can be 0 or 1, which represents  $\theta(z)$  as consisting of 1 or 2 terms, respectively. For each term,  $w_i$  is the weighting factor,  $p_i$  is the averaged number of daughter particles,  $q_i$  and  $r_i$  are the exponents, and  $\beta(q_i, r_i)$  is the beta function. The following constraints are imposed on the parameters in [Equation 2.26 \(p. 409\)](#) :

$$\sum_i w_i = 1 \quad (2.27)$$

$$\sum_i w_i p_i = p \quad (2.28)$$

$$\sum_i w_i \left( \frac{p_i q_i}{q_i + r_i} \right) = 1 \quad (2.29)$$

In order to demonstrate how to transform the generalized PDF to represent an appropriate daughter distribution, consider the expressions shown in the tables that follow:

**Table 2.3: Daughter Distributions**

Type	$\theta(z)$
Equal-size [14] (p. 459)	$p\delta(z - \frac{1}{p})$
Attrition [14] (p. 459)	$\delta(z - (1-\varepsilon)) + \delta(z - \varepsilon)$
Power Law [30] (p. 460)	$(\nu+1) z^{\nu-1}$
Parabolic -a [30] (p. 460)	$(\nu+2)(\nu+1)z^{\nu-1}(1-z)$
Austin [2] (p. 459)	$w(\nu_1+1) z^{\nu_1-1} + (1-w)(\nu_2+1) z^{\nu_2-1}$
Binary Beta -a [13] (p. 459)	$60z^2(1-z)^2$
Binary Beta -b [21] (p. 460)	$\frac{2}{\beta(\nu, \nu)} z^{\nu-1} (1-z)^{\nu-1}$
Uniform [30] (p. 460)	$p(p-1)(1-z)^{p-2}$

**Table 2.4: Daughter Distributions (cont.)**

Type	$p$	Constraints
Equal-size [14] (p. 459)	$p$	$p \geq 2$
Attrition [14] (p. 459)	2	$\varepsilon \ll 1$

Type	$p$	Constraints
Power Law [30] (p. 460)	$\frac{\nu+1}{\nu}$	$0 < \nu \leq 1$
Parabolic -a [30] (p. 460)	$\frac{\nu+2}{\nu}$	$0 < \nu \leq 2$
Austin [2] (p. 459)	$w\left(1+\frac{1}{\nu_1}\right) + (1-w)\left(1+\frac{1}{\nu_2}\right)$	$\nu_1, \nu_2 > 0; 1 \geq w \geq \nu_1 \left(\frac{\nu_2-1}{\nu_2-\nu_1}\right)$
Binary Beta -a [13] (p. 459)	2	N/A
Binary Beta -b [21] (p. 460)	2	$\nu > 0$
Uniform [30] (p. 460)	$p$	$p \geq 2$

In Table 2.3: Daughter Distributions (p. 409),  $\delta$  is the Dirac delta function,  $w$  is a weighting coefficient, and  $\varepsilon$ ,  $\nu$ ,  $\nu_1$ , and  $\nu_2$  are user-defined parameters.

The generalized form can represent the daughter distributions in Table 2.3: Daughter Distributions (p. 409) by using the values shown in Table 2.5: Values for Daughter Distributions in General Form (p. 410).

**Table 2.5: Values for Daughter Distributions in General Form**

Type	$w_0$	$p_0$	$q_0$	$r_0$	$w_1$	$p_1$	$q_1$	$r_1$	Constraints
Equal-size	1	$p$	$\infty^*$	$\infty^*$	N/A	N/A	N/A	N/A	$p \geq 2$
Attrition	0.5	2	$\varepsilon$	1	0.5	2	1	$\varepsilon$	$\varepsilon \ll 1$
Power Law	1	$\frac{\nu+1}{\nu}$	$\nu$	1	N/A	N/A	N/A	N/A	$0 < \nu \leq 1$
Parabolic	1	$\frac{\nu+2}{\nu}$	$\nu$	2	N/A	N/A	N/A	N/A	$0 < \nu \leq 2$
Austin	$w$	$\frac{\nu_1+1}{\nu_1}$	$\nu_1$	1	$1-w$	$\frac{\nu_2+1}{\nu_2}$	$\nu_2$	1	$\nu_1, \nu_2 > 0;$ $1 \geq w \geq \nu_1 \left(\frac{\nu_2-1}{\nu_2-\nu_1}\right)$
Binary Beta **	1	2	$\nu$	$\nu$	N/A	N/A	N/A	N/A	$\nu > 0$
Uniform	1	$p$	1	$p-1$	N/A	N/A	N/A	N/A	$p \geq 2$

(\*)You can approximate  $\infty$  by using a very large number, such as 1e20.

(\*\*)Binary Beta -a is a special case of Binary Beta -b when  $\nu=3$ .

### Important

Note that for the ANSYS Fluent implementation of the generalized form of the PDF, you will only enter values for  $w_0$ ,  $p_0$ ,  $q_0$ ,  $r_0$ , and  $q_1$ , and the remaining values ( $w_1$ ,  $p_1$ , and  $r_1$ ) will be calculated automatically.

### 2.2.2.7. Aggregation

The aggregation kernel [19] (p. 460) is expressed as

$$a(V, V')$$

The aggregation kernel has units of  $m^3/s$ , and is sometimes defined as a product of two quantities:

- the frequency of collisions between particles of volume  $V$  and particles of volume  $V'$
- the “efficiency of aggregation” (that is, the probability of particles of volume  $V$  coalescing with particles of volume  $V'$ ).

The birth rate of particles of volume  $V$  due to aggregation is given by

$$B_{ag} = \frac{1}{2} \int_0^V a(V-V', V') n(V-V') n(V') dV' \quad (2.30)$$

where particles of volume  $V-V'$  aggregate with particles of volume  $V'$  to form particles of volume  $V$ . The factor  $1/2$  is included to avoid accounting for each collision event twice.

The death rate of particles of volume  $V$  due to aggregation is given by

$$D_{ag} = \int_0^\infty a(V, V') n(V) n(V') dV' \quad (2.31)$$

### Important

The breakage and aggregation kernels depend on the nature of the physical application. For example, in gas-liquid dispersion, the kernels are functions of the local liquid-phase turbulent dissipation.

The following is a list of aggregation functions available in ANSYS Fluent:

- Constant
- Luo model
- Free molecular model
- Turbulent model
- User-defined model

The Luo, free molecular, and turbulent aggregation functions are described in detail in the sections that follow.

### 2.2.2.8. Luo Aggregation Kernel

For the Luo model [18] (p. 460), the general aggregation kernel is defined as the rate of particle volume formation as a result of binary collisions of particles with volumes  $V_i$  and  $V_j$ :

$$\Omega_{ag}(V_i, V_j) = \omega_{ag}(V_i, V_j) P_{ag}(V_i, V_j) [m^3 / sec] \quad (2.32)$$

where  $\omega_{ag}(V_i, V_j) [m^3 / sec]$  is the frequency of collision and  $P_{ag}(V_i, V_j)$  is the probability that the collision results in coalescence. The frequency is defined as follows:

$$\omega_{ag}(V_i, V_j) = \frac{\pi}{4} (d_i + d_j)^2 \bar{u}_{ij} \quad (2.33)$$

where  $\bar{u}_{ij}$  is the characteristic velocity of collision of two particles with diameters  $d_i$  and  $d_j$ .

$$\bar{u}_{ij} = (\bar{u}_i^2 + \bar{u}_j^2)^{1/2} \quad (2.34)$$

where

$$\bar{u}_i = 1.43(\varepsilon d_i)^{1/3} \quad (2.35)$$

The expression for the probability of aggregation is

$$P_{ag} = \exp \left\{ -c_1 \frac{[0.75(1+x_{ij}^2)(1+x_{ij}^3)]^{1/2}}{(\rho_2/\rho_1 + 0.5)^{1/2}(1+x_{ij})^3} We_{ij}^{1/2} \right\} \quad (2.36)$$

where  $c_1$  is a constant of order unity,  $x_{ij} = d_i/d_j$ ,  $\rho_1$  and  $\rho_2$  are the densities of the primary and secondary phases, respectively, and the Weber number is defined as

$$We_{ij} = \frac{\rho_i d_i (\bar{u}_{ij})^2}{\sigma} \quad (2.37)$$

### 2.2.2.9. Free Molecular Aggregation Kernel

Real particles aggregate and break with frequencies (or kernels) characterized by complex dependencies over particle internal coordinates [29] (p. 460). In particular, very small particles (say up to  $1\mu m$ ) aggregate because of collisions due to Brownian motions. In this case, the frequency of collision is size-dependent and usually the following kernel is implemented:

$$a(L_i, L_j) = \frac{2k_B T}{3\mu} \frac{(L_i + L_j)^2}{L_i L_j} \quad (2.38)$$

where  $k_B$  is the Boltzmann constant,  $T$  is the absolute temperature,  $\mu$  is the viscosity of the suspending fluid. This kernel is also known as the Brownian kernel or the perikinetic kernel.

### 2.2.2.10. Turbulent Aggregation Kernel

During mixing processes, mechanical energy is supplied to the fluid. This energy creates turbulence within the fluid. The turbulence creates eddies, which in turn help dissipate the energy. The energy is transferred from the largest eddies to the smallest eddies in which it is dissipated through viscous in-

teractions. The size of the smallest eddies is the Kolmogorov microscale,  $\eta$ , which is expressed as a function of the kinematic viscosity and the turbulent energy dissipation rate:

$$\eta = \left( \frac{v^3}{\epsilon} \right)^{1/4} \quad (2.39)$$

In the turbulent flow field, aggregation can occur by two mechanisms:

- viscous subrange mechanism: this is applied when particles are smaller than the Kolmogorov microscale,  $\eta$
- inertial subrange mechanism: this is applied when particles are bigger than the Kolmogorov microscale. In this case, particles assume independent velocities.

For the viscous subrange, particle collisions are influenced by the local shear within the eddy. Based on work by Saffman and Turner [28] (p. 460), the collision rate is expressed as,

$$a(L_i, L_j) = \zeta_T \sqrt{\frac{8\pi}{15}} \dot{\gamma} \frac{(L_i + L_j)^3}{8} \quad (2.40)$$

where  $\zeta_T$  is a pre-factor that takes into account the capture efficiency coefficient of turbulent collision, and  $\dot{\gamma}$  is the shear rate:

$$\dot{\gamma} = \frac{\epsilon}{v^{0.5}} \quad (2.41)$$

For the inertial subrange, particles are bigger than the smallest eddy, therefore they are dragged by velocity fluctuations in the flow field. In this case, the aggregation rate is expressed using Abrahamson's model [1] (p. 459),

$$a(L_i, L_j) = \zeta_T 2^{3/2} \sqrt{\pi} \frac{(L_i + L_j)^2}{4} \sqrt{(U_i^2 + U_j^2)} \quad (2.42)$$

where  $U_i^2$  is the mean squared velocity for particle  $i$ .

The empirical capture efficiency coefficient of turbulent collision describes the hydrodynamic and attractive interaction between colliding particles. Higashitani et al. [11] (p. 459) proposed the following relationship:

$$\zeta_T = 0.732 \left( \frac{5}{N_T} \right)^{0.242}; N_T \geq 5 \quad (2.43)$$

where  $N_T$  is the ratio between the viscous force and the Van der Waals force,

$$N_T = \frac{6\pi\mu(L_i + L_j)^3 \lambda}{8H} \quad (2.44)$$

Where  $H$  is the Hamaker constant, a function of the particle material, and  $\lambda$  is the deformation rate,

$$\lambda = \left( \frac{4\epsilon}{15\pi v} \right)^{0.5} \quad (2.45)$$

### 2.2.3. Particle Birth by Nucleation

Depending on the application, spontaneous nucleation of particles can occur due to the transfer of molecules from the primary phase. For example, in crystallization from solution, the first step is the phase separation or "birth" of new crystals. In boiling applications, the creation of the first vapor bubbles is a nucleation process referred to as nucleate boiling.

The nucleation rate is defined through a boundary condition as shown in [Equation 2.5 \(p. 404\)](#).

## 2.3. Solution Methods

As discussed in [Introduction \(p. 399\)](#), the population balance equation can be solved by the four different methods in ANSYS Fluent: the discrete method, the inhomogeneous discrete method, the standard method of moments (SMM), and the quadrature method of moments (QMOM). For each method, the ANSYS Fluent implementation is limited to a single internal coordinate corresponding to particle size. The following subsections describe the theoretical background of each method and list their advantages and disadvantages.

[2.3.1. The Discrete Method and the Inhomogeneous Discrete Method](#)

[2.3.2. The Standard Method of Moments \(SMM\)](#)

[2.3.3. The Quadrature Method of Moments \(QMOM\)](#)

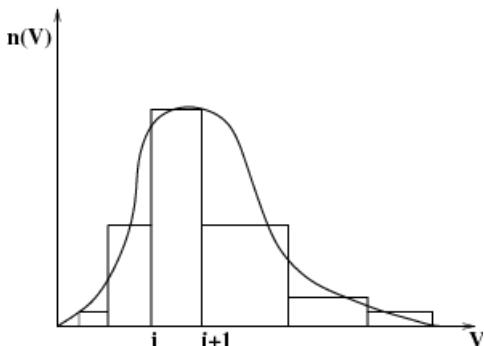
[2.3.4. The Direct Quadrature Method of Moments \(DQMOM\)](#)

### 2.3.1. The Discrete Method and the Inhomogeneous Discrete Method

The discrete method (also known as the classes or sectional method) was developed by Hounslow [12] (p. 459), Litster [17] (p. 459), and Ramkrishna [26] (p. 460). It is based on representing the continuous particle size distribution (PSD) in terms of a set of discrete size classes or bins, as illustrated in [Figure 2.1: A Particle Size Distribution as Represented by the Discrete Method \(p. 414\)](#). The advantages of this method are its robust numerics and that it gives the PSD directly. The disadvantages are that the bins must be defined *a priori* and that a large number of classes may be required.

The solution methods for the inhomogeneous discrete method are based on the discrete method and therefore share many of the same fundamentals.

**Figure 2.1: A Particle Size Distribution as Represented by the Discrete Method**



#### 2.3.1.1. Numerical Method

In ANSYS Fluent, the PBE is written in terms of volume fraction of particle size  $i$ :

$$\frac{\partial}{\partial t} \left( \rho_s \alpha_i \right) + \nabla \cdot \left( \rho_s u_i \alpha_i \right) + \frac{\partial}{\partial V} \left( \frac{G_v \rho_s \alpha_i}{V} \right) = \rho_s V_i \left( B_{ag,i} - D_{ag,i} + B_{br,i} - D_{br,i} \right) + 0^i \rho_s V_0 \dot{n}_0 \quad (2.46)$$

where  $\rho_s$  is the density of the secondary phase and  $\alpha_i$  is the volume fraction of particle size  $i$ , defined as

$$\alpha_i = N_i V_i \quad i=0,1,\dots,N-1 \quad (2.47)$$

where

$$N_i(t) = \int_{V_i}^{V_{i+1}} n(V, t) dV \quad (2.48)$$

and  $V_i$  is the volume of the particle size  $i$ . In ANSYS Fluent, a fraction of  $\alpha$ , called  $f_i$ , is introduced as the solution variable. This fraction is defined as

$$f_i = \frac{\alpha_i}{\alpha} \quad (2.49)$$

where  $\alpha$  is the total volume fraction of the secondary phase.

The nucleation rate  $n_0$  appears in the discretized equation for the volume fraction of the smallest size  $V_0$ . The notation  $0^i$  signifies that this particular term, in this case  $\rho_s V_0 n_0$ , appears in [Equation 2.46 \(p. 414\)](#) only in the case of the smallest particle size.

The growth rate in is discretized as follows [Equation 2.46 \(p. 414\) \[12\] \(p. 459\)](#):

$$\frac{\partial}{\partial V} \left( \frac{G_v \rho_s \alpha_i}{V} \right) = \rho_s V_i \left[ \left( \frac{G_{v,i-1} N_{i-1}}{V_i - V_{i-1}} \right) - \left( \frac{G_{v,i} N_i}{V_{i+1} - V_i} \right) \right] \quad (2.50)$$

The volume coordinate is discretized as [\[12\] \(p. 459\)](#)  $V_{i+1}/V_i = 2^q$  where  $q=1, 2, \dots$  and is referred to as the "ratio factor".

The particle birth and death rates are defined as follows:

$$B_{ag,i} = \sum_{k=1}^N \sum_{j=1}^N a_{kj} N_k N_j x_{kj} \xi_{kj} \quad (2.51)$$

$$D_{agi} = \sum_{j=1}^N a_{ij} N_i N_j \quad (2.52)$$

$$B_{br,i} = \sum_{j=i+1}^N g(V_j) N_j \beta(V_i | V_j) \quad (2.53)$$

$$D_{br,i} = g(V_i) N_i \quad (2.54)$$

where  $a_{ij} = a(V_i, V_j)$  and

$$\xi_{kj} = \begin{cases} 1 & \text{for } V_i < V_{ag} < V_{i+1}, \text{ where } i \leq N-1 \\ 0 & \text{otherwise} \end{cases} \quad (2.55)$$

$V_{ag}$  is the particle volume resulting from the aggregation of particles  $k$  and  $j$ , and is defined as

$$V_{ag} = [x_{kj} V_i + (1-x_{kj}) V_{i+1}] \quad (2.56)$$

where

$$x_{kj} = \frac{V_{ag} - V_{i+1}}{V_i - V_{i+1}} \quad (2.57)$$

If  $V_{ag}$  is greater than or equal to the largest particle size  $V_N$ , then the contribution to class  $N-1$  is

$$x_{kj} = \frac{V_{ag}}{V_N} \quad (2.58)$$

**Important**

Note that there is no breakage for the smallest particle class.

**2.3.1.2. Breakage Formulations for the Discrete Method**

The default breakage formulation for the discrete method in ANSYS Fluent is based on the Hagesather method [10] (p. 459). In this method, the breakage sources are distributed to the respective size bins, preserving mass and number density. For the case when the ratio between successive bin sizes can be expressed as  $2^n$  where  $n=1, 2, \dots$ , the source in bin  $i$ , ( $i=1, \dots, N$ ) can be expressed as

$$\begin{aligned} B_b(i) &= \sum_{k=i+1, k \neq N}^N \Omega_b(v_k, v_i) + \sum_{k=i, k \neq N}^i x_{i+1,k} \Omega_b(v_{i+1}, v_k) \\ &+ \sum_{k=1, k \neq 1}^{i-1} (1-x_{i,k}) g(v_{i+1} \Omega_b(v_i, v_k)) \end{aligned} \quad (2.59)$$

Here

$$\Omega_b(v_k, v_i) = N_k g(v_k) \beta(v_k, v_i) \quad (2.60)$$

A more mathematically rigorous formulation is given by Ramkrishna [15] (p. 459), where the breakage rate is expressed as

$$B_b(i) = \sum_i^N n_{i,k} g(v_k) N_k \quad (2.61)$$

where

$$n_{i,k} = \int_{v_i}^{v_{i+1}} \frac{v_{i+1}-v}{v_{i+1}-v_i} \beta(v_k, v) dv + \int_{v_{i-1}}^{v_i} \frac{v-v_{i-1}}{v_i-v_{i-1}} \beta(v_k, v) dv \quad (2.62)$$

The Ramkrishna formulation can be slow due to the large number of integration points required. However, for simple forms of  $\beta$ , the integrations can be performed relatively easily. The Hagesather formulation requires fewer integration points and the difference in accuracy with the Ramkrishna formulation can be corrected by a suitable choice of bin sizes.

**Important**

To keep the computing time reasonable, a volume averaged value is used for the turbulent eddy dissipation when the Luo model is used in conjunction with the Ramkrishna formulation.

**Note**

The inhomogeneous discrete phase applies the Hagesather formulation.

## 2.3.2. The Standard Method of Moments (SMM)

The SMM, proposed by Randolph and Larson [27] (p. 460) is an alternative method for solving the PBE. Its advantages are that it reduces the dimensionality of the problem and that it is relatively simple to solve transport equations for lower-order moments. The disadvantages are that exact closure of the right-hand side is possible only in cases of constant aggregation and size-independent growth, and that breakage modeling is not possible. The closure constraint is overcome, however, through QMOM (see [The Quadrature Method of Moments \(QMOM\) \(p. 418\)](#)).

### 2.3.2.1. Numerical Method

The SMM approach is based on taking moments of the PBE with respect to the internal coordinate (in this case, the particle size  $L$ ).

Defining the  $k^{\text{th}}$  moment as

$$m_k(\vec{x}, t) = \int_0^\infty n(L; \vec{x}, t) L^k dL \quad k=0, 1, \dots, N-1 \quad (2.63)$$

and assuming constant particle growth, its transport equation can be written as

$$\frac{\partial}{\partial t} (\rho_s m_k) + \nabla \cdot (\rho_s \vec{u} m_k) = \rho_s (\bar{B}_{ag,k} - \bar{D}_{ag,k} + \bar{B}_{br,k} - \bar{D}_{br,k}) + \rho_s (0^k \dot{n}_0 + \text{Growth}) \quad (2.64)$$

where

$$\bar{B}_{ag,k} = \frac{1}{2} \int_0^\infty n(\lambda) \int_0^\infty a(u, \lambda)(u, \lambda) (u^3 + \lambda^3)^{k/3} n(u) du d\lambda \quad (2.65)$$

$$\bar{D}_{ag,k} = \int_0^\infty L^k n(L) \int_0^\infty a(L, \lambda) n(\lambda) d\lambda dL \quad (2.66)$$

$$\bar{B}_{br,k} = \int_0^\infty L^k \int_0^\infty g(\lambda) \beta(L|\lambda) n(\lambda) d\lambda dL \quad (2.67)$$

$$\bar{D}_{br,k} = \int_0^k L^k g(L) n(L) dL \quad (2.68)$$

$N$  is the specified number of moments and  $\dot{n}_0$  is the nucleation rate. The growth term is defined as

$$\text{Growth} \equiv \int_0^\infty k L^{k-1} G(L) n(L, t) dL \quad (2.69)$$

and for constant growth is represented as

$$k G m_{k-1} \quad (2.70)$$

Equation 2.65 (p. 417) can be derived by using

$$u^3 = L^3 - \lambda^3; dL = \frac{u^2}{L^2} du$$

and reversing the order of integration. From these moments, the parameters describing the gross properties of particle population can be derived as

$$N_{total} = m_0 \quad (2.71)$$

$$L_{total} = m_1 \quad (2.72)$$

$$A_{total} = K_a m_2 \quad (2.73)$$

$$V_{total} = K_v m_3 \quad (2.74)$$

$$d_{32} = \frac{m_3}{m_2} \quad (2.75)$$

These properties are related to the total number, length, area, and volume of solid particles per unit volume of mixture suspension. The Sauter mean diameter,  $d_{32}$ , is usually used as the mean particle size.

To close [Equation 2.64 \(p. 417\)](#), the quantities represented in [Equation 2.65 \(p. 417\) – Equation 2.68 \(p. 417\)](#) need to be expressed in terms of the moments being solved. To do this, one approach is to assume size-independent kernels for breakage and aggregation, in addition to other simplifications such as the Taylor series expansion of the term  $(u^3 + \lambda^3)^{k/3}$ . Alternatively, a profile of the PSD could be assumed so that [Equation 2.65 \(p. 417\) – Equation 2.68 \(p. 417\)](#) can be integrated and expressed in terms of the moments being solved.

In ANSYS Fluent, an exact closure is implemented by restricting the application of the SMM to cases with size-independent growth and a constant aggregation kernel.

### 2.3.3. The Quadrature Method of Moments (QMOM)

The quadrature method of moments (QMOM) was first proposed by McGraw [\[22\] \(p. 460\)](#) for modeling aerosol evolution and coagulation problems. Its applications by Marchisio et al. [\[20\] \(p. 460\)](#) have shown that the method requires a relatively small number of scalar equations to track the moments of population with small errors.

The QMOM provides an attractive alternative to the discrete method when aggregation quantities, rather than an exact PSD, are desired. Its advantages are fewer variables (typically only six or eight moments) and a dynamic calculation of the size bins. The disadvantages are that the number of abscissas may not be adequate to describe the PSD and that solving the Product-Difference algorithm may be time consuming.

#### 2.3.3.1. Numerical Method

The quadrature approximation is based on determining a sequence of polynomials orthogonal to  $n(L)$  (that is, the particle size distribution). If the abscissas of the quadrature approximation are the nodes of the polynomial of order  $N$ , then the quadrature approximation

$$\int_0^\infty f(L)n(L)dL \approx \sum_{i=1}^N f(L_i)w_i, \quad (2.76)$$

is exact if  $f(L)$  is a polynomial of order  $N$  or smaller [\[5\] \(p. 459\)](#). In all other cases, the closer  $f(L)$  is to a polynomial, the more accurate the approximation.

A direct way to calculate the quadrature approximation is by means of its definition through the moments:

$$m_k = \sum_{i=1}^N w_i L_i^k. \quad (2.77)$$

The quadrature approximation of order  $N$  is defined by its  $N$  weights  $w_i$  and  $N$  abscissas  $L_i$  and can be calculated by its first  $2N$  moments  $m_0, \dots, m_{2N-1}$  by writing the recursive relationship for the poly-

nomials in terms of the moments  $m_k$ . Once this relationship is written in matrix form, it is easy to show that the roots of the polynomials correspond to the eigenvalues of the Jacobi matrix [25] (p. 460). This procedure is known as the Product-Difference algorithm [9] (p. 459). Once the weights and abscissas are known, the source terms due to coalescence and breakage can be calculated and therefore the transport equations for the moments can be solved.

Applying [Equation 2.76 \(p. 418\)](#) and [Equation 2.77 \(p. 418\)](#), the birth and death terms in [Equation 2.64 \(p. 417\)](#) can be rewritten as

$$\bar{B}_{ag,k} = \frac{1}{2} \sum_{i=1}^N w_i \sum_{j=1}^N w_j (L_i^3 + L_j^3)^{k/3} a(L_i, L_j) \quad (2.78)$$

$$\bar{D}_{ag,k} = \sum_{i=1}^N L_i^k w_i \sum_{j=1}^N w_j a(L_i, L_j) \quad (2.79)$$

$$\bar{B}_{br,k} = \sum_{i=1}^N w_i \int_0^\infty L_k g(L_i) \beta(L|L_i) dL \quad (2.80)$$

$$\bar{D}_{br,k} = \sum_{i=1}^N w_i L_i^k g(L_i) \quad (2.81)$$

Theoretically, there is no limitation on the expression of breakage and aggregation kernels when using QMOM.

The nucleation rate is defined in the same way as for the SMM. The growth rate for QMOM is defined by [Equation 2.69 \(p. 417\)](#) and represented as

$$\sum_{i=1}^N w_i L_i^{k-1} G(L_i) \quad (2.82)$$

to allow for a size-dependent growth rate.

### 2.3.4. The Direct Quadrature Method of Moments (DQMOM)

DQMOM equations are derived from the basic number density function equation via the moment transfer method, in a similar way to QMOM. The difference is that DQMOM assumes that each Quadrature point will occupy an independent velocity field, whereas QMOM assumes that all Quadrature points are moving on the same velocity field. This difference enables DQMOM to predict particle segregation due to particle interaction.

In this implementation of DQMOM, four phases must be specified: one primary phase and three secondary phases that are DQMOM phases. Compared to QMOM, for a three Quadrature points system, the DQMOM method only needs three extra equations to solve for the effective length of the particle, but there are additional source terms for the volume fraction equation for each DQMOM phase. In ANSYS Fluent, three particle interactions are accounted for, growth, aggregation, and breakage. Nucleation is not considered.

#### 2.3.4.1. Numerical Method

The DQMOM equations, describing a poly-dispersed particle system undergoing aggregation, breakage, and growth can be written as follows (the details of the DQMOM formulation can be found in [7] (p. 459)):

$$\frac{\partial \varepsilon_i \rho_s}{\partial t} + \nabla \cdot (\vec{u}_{si} \varepsilon_i \rho_s) = 3k_v \rho_s L_i^2 (b_i + w_i G_i) - 2k_v \rho_s L_i^3 a_i \quad (2.83)$$

$$\frac{\partial \varepsilon_i L_i \rho_s}{\partial t} + \nabla \cdot (\bar{u}_s \varepsilon_i L_i \rho_s) = 4k_v \rho_s L_i^3 (b_i + w_i G_i) - 3k_v \rho_s L_i^4 a_i \quad (2.84)$$

where  $\varepsilon_i$  and  $\varepsilon_i L_i$  are the VOF and the effective length of the particle phase, respectively.  $w_i$  is the number of particles per unit volume and  $G_i$  is the growth rate at Quadrature point  $i$ , while  $a_i$  and  $b_i$  can be computed through a linear system resulting from the moment transformation of the particle number density transport equation using  $N$  Quadrature points. The linear system can be written in matrix form as

$$A\alpha = d \quad (2.85)$$

Where the  $2N \times 2N$  coefficient matrix  $A = [A_1 \ A_2]$  is defined by

$$A_1 = \begin{bmatrix} 1 & \dots & 1 \\ 0 & \dots & 0 \\ -L_1^2 & \dots & -L_N^2 \\ \vdots & \ddots & \vdots \\ 2(1-N)L_1^{2N-1} & \dots & 2(1-N)L_N^{2N-1} \end{bmatrix} \quad (2.86)$$

$$A_2 = \begin{bmatrix} 0 & \dots & 0 \\ 1 & \dots & 1 \\ 2L_1 & \dots & L_N \\ \vdots & \ddots & \vdots \\ (2N-1)L_1^{2N-2} & \dots & (2N-1)L_N^{2N-2} \end{bmatrix} \quad (2.87)$$

The  $2N$  vector of unknowns  $\alpha$  is defined by

$$\alpha = [a_1 \ \dots \ a_N \ b_1 \ \dots \ b_N]^T = \begin{bmatrix} a \\ b \end{bmatrix} \quad (2.88)$$

The right hand side of [Equation 2.85 \(p. 420\)](#) is the known source terms involving aggregation and breakage phenomena only. The growth term is accounted for directly in [Equation 2.83 \(p. 419\)](#) and [Equation 2.84 \(p. 420\)](#). At present, nucleation is not considered.

$$d = [S_0^{(N)} \ \dots \ S_{2N-1}^{(N)}]^T \quad (2.89)$$

The source term for the  $k^{th}$  moment  $S_k^{(N)}$  ( $k=0, \dots, 2N-1$ ) is defined as

$$S_k^{(N)}(x, t) = \int_0^\infty L^k S(x, t) dL \quad (2.90)$$

When the abscissas of the Quadrature points  $L_i$  are distinct, the matrix  $A$  is well defined and a unique solution of [Equation 2.85 \(p. 420\)](#) can be obtained. Otherwise, the matrix  $A$  is not full rank and cannot be inverted to find a unique solution for  $\alpha$ . The method adopted by ANSYS Fluent to overcome this problem is to employ a perturbation technique. For example, for the current three Quadrature points system, the perturbation technique will add a small value to the abscissas to make sure the matrix  $A$  is full rank. It is important to note that the perturbation technique is only used for the definition of matrix  $A$  and no modifications are made to the source term vector of [Equation 2.90 \(p. 420\)](#). Therefore, both the weights and overall source terms resulting from aggregation and breakage are not affected by the perturbation method. The simulation tests have found that the perturbation method can stabilize the solutions of [Equation 2.85 \(p. 420\)](#) and reduce the physically unrealistically large source terms for the two phases whose abscissa are too close in value. However, the technique has little effect on the phase whose abscissa  $L_i$  is distinct from the other two.

## 2.4. Population Balance Statistics

The following sections introduce statistical concepts that are useful when using the population balance models.

### 2.4.1. Reconstructing the Particle Size Distribution from Moments

#### 2.4.2. The Log-Normal Distribution

### 2.4.1. Reconstructing the Particle Size Distribution from Moments

Given a set of moments, the most likely PSD can be obtained based on the "statistically most probable" distribution for turbulent flames [24] (p. 460), which was adapted for crystallization problems by Baldyga and Orciuch [3] (p. 459).

The number density function  $n(L)$  is expressed as

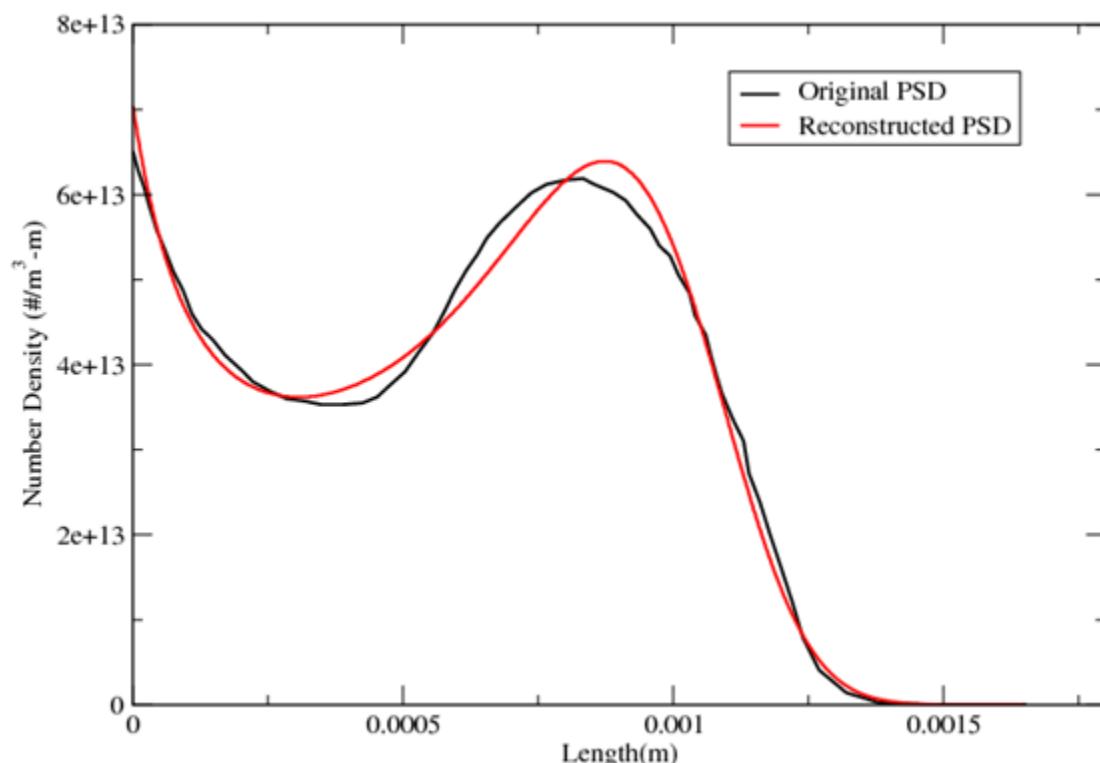
$$n(L) = \exp\left(\sum_{i=0}^{N-1} A_i L^i\right) \quad (2.91)$$

The equation for the  $k^{\text{th}}$  moment is now written as

$$m_k = \int_0^{\infty} L^k \exp\left(\sum_{i=0}^{N-1} A_i L^i\right) dL \quad k=0, 1, \dots, N-1 \quad (2.92)$$

Given  $N$  moments, the coefficients  $A_i$  can be found by a globally convergent Newton-Raphson method to reconstruct the particle size distribution (for example, [Figure 2.2: Reconstruction of a Particle Size Distribution \(p. 421\)](#)).

**Figure 2.2: Reconstruction of a Particle Size Distribution**



## 2.4.2. The Log-Normal Distribution

When using either of the discrete population balance methods, you have the option of specifying the size distribution at a velocity inlet by specifying a log-normal distribution.

The log-normal distribution for the number density,  $n$ , as a function of the particle size,  $L$ , can be written as:

$$n(L) = \frac{1}{L\sigma\sqrt{2\pi}} e^{\frac{(\ln L - \mu)^2}{2\sigma^2}} \quad (2.93)$$

where  $\mu$  and  $\sigma$  are, respectively, the location and scale parameters of the distribution and can be written as

$$\begin{aligned}\mu &= \ln(\mu') - \frac{1}{2} \ln\left(1 + \frac{\sigma'^2}{\mu'^2}\right) \\ \sigma^2 &= \ln\left(1 + \frac{\sigma'^2}{\mu'^2}\right)\end{aligned}$$

$\mu'$  and  $\sigma'$  are the mean and standard deviation, respectively, and are specified in the boundary conditions as shown in [Initializing Bin Fractions With a Log-Normal Distribution \(p. 436\)](#).

---

## Chapter 3: Using the ANSYS Fluent Population Balance Model

---

This chapter provides basic instructions to install the population balance model and solve population balance problems in ANSYS Fluent. It assumes that you are already familiar with standard ANSYS Fluent features, including the user-defined function procedures described in the [Fluent Customization Manual](#). This chapter describes the following:

- 3.1. Population Balance Module Installation
- 3.2. Loading the Population Balance Module
- 3.3. Population Balance Model Setup

### 3.1. Population Balance Module Installation

The population balance module is provided as an add-on module with the standard ANSYS Fluent licensed software.

### 3.2. Loading the Population Balance Module

The population balance module is loaded into ANSYS Fluent through the text user interface (TUI). The module can only be loaded when a valid ANSYS Fluent case file has been set or read. The text command to load the module is:

```
define → models → addon-module
```

A list of ANSYS Fluent add-on modules is displayed:

```
> /define/models/addon-module
Fluent Addon Modules:
 0. None
 1. MHD Model
 2. Fiber Model
 3. Fuel Cell and Electrolysis Model
 4. SOFC Model with Unresolved Electrolyte
 5. Population Balance Model
 6. Adjoint Solver
 7. Single-Potential Battery Model
 8. Dual-Potential MSMD Battery Model
 9. PEM Fuel Cell Model
Enter Module Number: [0] 5
```

Select the Population Balance Model by entering the module number 5. During the loading process a scheme library containing the graphical and text user interface, and a UDF library containing a set of user defined functions are loaded into ANSYS Fluent. A message **Addon Module: pop-bal...loaded!** is displayed at the end of the loading process.

The population balance module setup is saved with the ANSYS Fluent case file. The module is loaded automatically when the case file is subsequently read into ANSYS Fluent. Note that in the saved case file, the population balance module is saved with the absolute path. Therefore, if the locations of the population balance module installation or the saved case file are changed, ANSYS Fluent will not be able to load the module when the case file is subsequently read.

### 3.3. Population Balance Model Setup

Following the loading of the population balance module, enable either the mixture or Eulerian multiphase model. This will enable you to activate the population balance model, where you will specify the appropriate parameters, and supply multiphase boundary conditions. These inputs are described in this chapter. Using the double-precision version of ANSYS Fluent when solving population balance problems is highly recommended.

---

#### Important

A limitation of the population balance model is that it can be used only on one secondary phase, even if your problem includes additional secondary phases. Note that a three-phase gas-liquid-solid case can be modeled, where the population balance model is used for the gas phase and the solid phase acts as a catalyst. However, if you are using the **Inhomogeneous Discrete**, more than one secondary phase can be used. Note that the properties of the secondary phases selected for that method should be the same for consistency.

---

For more information, see the following sections:

- 3.3.1. Enabling the Population Balance Model
- 3.3.2. Defining Population Balance Boundary Conditions
- 3.3.3. Specifying Population Balance Solution Controls
- 3.3.4. Coupling With Fluid Dynamics
- 3.3.5. Specifying Interphase Mass Transfer Due to Nucleation and Growth

#### 3.3.1. Enabling the Population Balance Model

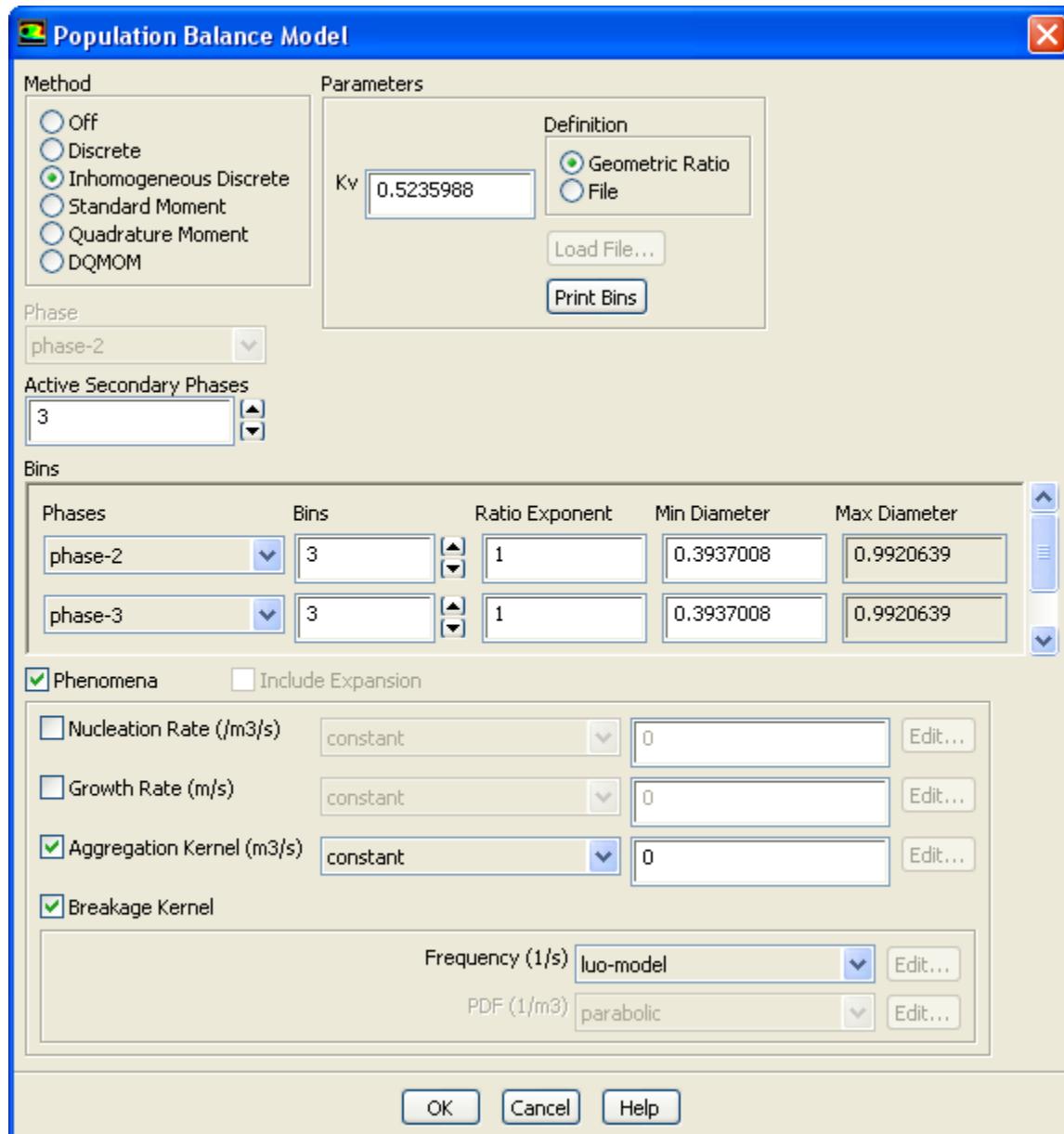
The procedure for setting up a population balance problem is described below. (Note that this procedure includes only those steps necessary for the population balance model itself; you will need to set up other models, boundary conditions, and so on, as usual. See the [ANSYS Fluent User's Guide](#) for details.)

1. Start the double-precision version of ANSYS Fluent.
2. To enable the population balance model, follow the instructions in [Loading the Population Balance Module \(p. 423\)](#).

Remember to enable the mixture or Eulerian multiphase model.

3. Open the **Population Balance Model** dialog box ([Figure 3.1:The Population Balance Model Dialog Box \(p. 425\)](#)).

 **Setup** → **Models** → **Population Balance**  **Edit...**

**Figure 3.1: The Population Balance Model Dialog Box**

**4. Specify the population balance method under **Method**.**

- If you select **Discrete**, you will need to specify the following parameters:

#### K<sub>v</sub>

specifies the value for the particle volume coefficient  $K_v$  (as described in [Particle Growth and Dissolution \(p. 404\)](#)). By default, this coefficient has a value of  $\pi / 6$ .

#### Definition

can be specified as a **Geometric Ratio** or as a **File**. If **Geometric Ratio** is selected, then the **Ratio Exponent** must be specified. If **File** is selected, you will click the **Load File...** button and select the bin size file that you want loaded.

You can input the diameter through the text file, with each diameter listed on a separate line, starting from the smallest to the largest diameter (one entry per line). Hence, you are not limited by the choices specified in the dialog box.

**Bins**

specifies the number of particle size bins used in the calculation.

**Ratio Exponent**

specifies the exponent  $q$  used in the discretization of the growth term volume coordinate (see [Numerical Method \(p. 414\)](#)).

**Min Diameter**

specifies the minimum bin size  $L_0 \equiv (V_0 / K_v)^{1/3}$ .

**Max Diameter**

displays the maximum bin size, which is calculated internally.

To display a list of the bin sizes in the console window, click **Print Bins**. The bin sizes will be listed in order of size, from the largest to the smallest. This option is only available when the **Geometric Ratio Definition** is selected.

- If you select **Inhomogeneous Discrete** under **Method**, you will specify the same parameters as for the **Discrete** model. Additionally, you can include more than one secondary phase in the bin definition. Enter the total number of **Active Secondary Phases** in your simulation.

---

**Note**

While reading bins through the **Load File...** option for the **Inhomogeneous Discrete** model, the corresponding phase name must be included, for example (( "air-1" (0.1 0.2 0.3))

---

- If you select **Standard Moment** under **Method**, you will specify the number of **Moments** under **Parameters**.
- If you selected **Quadrature Moment** under **Method**, you will set the number of moments to either 4, 6 or 8 under **Parameters**.
- If you selected **DQMOM** under **Method**, you will select the **DQMOM Phases** from the list. You will also specify the following **Parameters**:

**Max Size**

specifies the maximum size of the particle.

**Min Size**

specifies the minimum size of the particle.

**Reference Length**

is the reference particle size. Normally, the averaged size of the particle group should be sufficient.

**Min VOF**

is the minimum VOF, where the total volume fraction of the particle phases (participating in DQMOM computations) is below the minimum value; the source terms caused by the breakage and coalescence are not computed in that cell for the DQMOM and VOF equations.

**Max VOF Change/Time Step**

is the maximum VOF change in percentage for each DQMOM phase per time step, in order to smooth the convergence progress.

**Generate DQMOM Values**

enables you to generate DQMOM values from PDF, CDF, or Overall Moments files. Each of the file formats and the way to generate the values are discussed in [Generated DQMOM Values \(p. 432\)](#).

**Note**

The DQMOM method is restricted to a four-phase system, of which three secondary phases are directly involved in the DQMOM computation. Unsteady simulations are required to model breakage and coalescence and a well defined initial field is recommended, in which the abscissas are distinct. Only growth, breakage, and aggregation are the available phenomena. No nucleation is considered.

5. Select the secondary phase from the **Phase** drop-down list for which you want to apply the population balance model parameters.
6. For all population balance methods, you can enable the following under **Phenomena** :

**Nucleation Rate**

enables you to specify the nucleation rate (particles/m<sup>3</sup>-s). You can select **constant** or **user-defined** from the drop-down list. If you select **constant**, specify a value in the adjacent field. If you have a user-defined function (UDF) that you want to use to model the nucleation rate, you can choose the **user-defined** option and specify the appropriate UDF.

**Note**

This option is not available when using the **Inhomogeneous Discrete** method.

**Growth Rate**

enables you to specify the particle growth rate (m/s). You can select **constant** or **user-defined** from the drop-down list. If you select **constant**, specify a value in the adjacent field. If you have a user-defined function (UDF) that you want to use to model the growth rate, you can choose the **user-defined** option and specify the appropriate UDF.

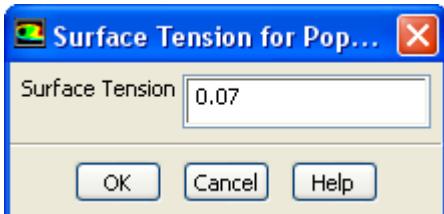
**Note**

This option is not available when using the **Inhomogeneous Discrete** method.

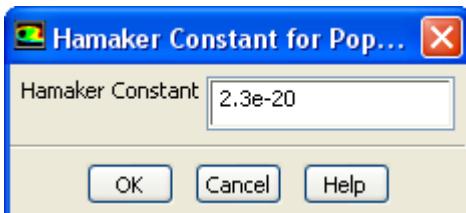
**Aggregation Kernel**

enables you to specify the aggregation kernel (m<sup>3</sup>/s). You can select **constant**, **luo-model**, **free-molecular-model**, **turbulent-model**, or **user-defined** from the drop-down list:

- If you select **constant**, specify a value in the adjacent field.
- If you select **luo-model**, the **Surface Tension for Population Balance** dialog box will open automatically to enable you to specify the surface tension (see [Figure 3.2: The Surface Tension for Population Balance Dialog Box \(p. 428\)](#)). The aggregation rate for the model will then be calculated based on Luo's aggregation kernel (as described in [Particle Birth and Death Due to Breakage and Aggregation \(p. 404\)](#)).

**Figure 3.2: The Surface Tension for Population Balance Dialog Box**

- If you select **free-molecular-model**, then [Equation 2.38 \(p. 412\)](#) is applied.
- If you select **turbulent-model**, the **Hamaker Constant for Population Balance** dialog box will open automatically to enable you to specify the Hamaker constant (see [Figure 3.3: The Hamaker Constant for Population Balance Dialog Box \(p. 428\)](#)). More information about this model is available in [Turbulent Aggregation Kernel \(p. 412\)](#).

**Figure 3.3: The Hamaker Constant for Population Balance Dialog Box**

- If you have a user-defined function (UDF) that you want to use to model the aggregation rate, you can choose the **user-defined** option and specify the appropriate UDF.

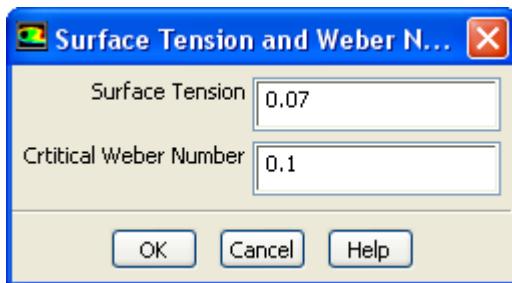
### **Breakage Kernel**

enables you to specify the particle breakage frequency ( $\text{particles}/\text{m}^3\text{-s}$ ). You can select **constant**, **luo-model**, **lehr-model**, **ghadiri-model**, **laakkonen-model** or **user-defined** from the **Frequency** drop-down list:

- If you select **constant**, specify a value in the adjacent field.
- If you select **luo-model**, the **Surface Tension for Population Balance** dialog box will open automatically to enable you to specify the surface tension (see [Figure 3.2: The Surface Tension for Population Balance Dialog Box \(p. 428\)](#)). The frequency used in the breakage rate will then be calculated based on Luo's breakage kernel (as described in [Particle Birth and Death Due to Breakage and Aggregation \(p. 404\)](#)).
- If you select **lehr-model**, the **Surface Tension and Weber Number** dialog box will open automatically to enable you to specify the surface tension and critical Weber number (see [Figure 3.4: The Surface Tension and Weber Number Dialog Box \(p. 429\)](#)). The frequency used in the breakage rate will then

be calculated based on Lehr's breakage kernel (as described in [Particle Birth and Death Due to Breakage and Aggregation \(p. 404\)](#)).

**Figure 3.4: The Surface Tension and Weber Number Dialog Box**



- If you select **ghadiri-model**, the **Ghadiri Breakage Constant for Population Balance** dialog box will open automatically to enable you to specify the breakage constant (see [Figure 3.5: The Ghadiri Breakage Constant for Population Balance Dialog Box \(p. 429\)](#)). The frequency will then be calculated based on Ghadiri's breakage kernel (as described in [Particle Birth and Death Due to Breakage and Aggregation \(p. 404\)](#)).

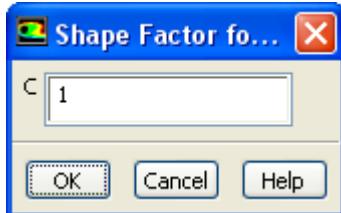
**Figure 3.5: The Ghadiri Breakage Constant for Population Balance Dialog Box**



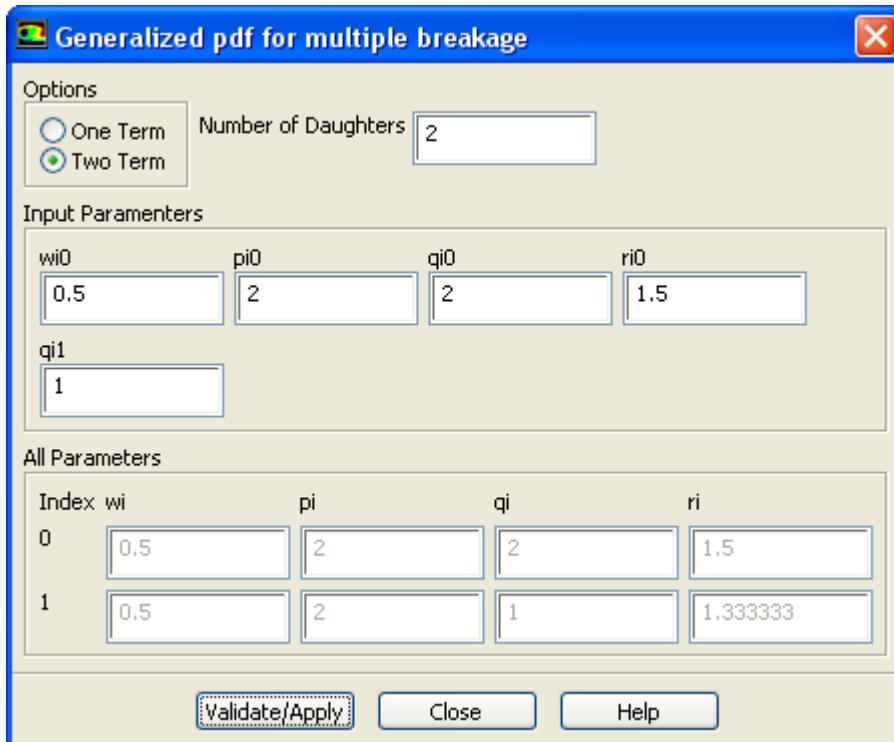
- If you select **laakkonen-model**, the **Surface Tension for Population Balance** dialog box will open automatically to enable you to specify the **Surface Tension** and the constant **C2** ([Laakkonen Breakage Kernels \(p. 407\)](#)). The frequency will then be calculated based on Laakkonen's breakage kernel (as described in [Laakkonen Breakage Kernels \(p. 407\)](#)).
- If you have a user-defined function (UDF) that you want to use to model the frequency for the breakage rate, you can choose the **user-defined** option and specify the appropriate UDF.

If you selected **constant**, **ghadiri-model**, **laakkonen-model**, or **user-defined** for **Frequency**, then you can specify the probability density function used to calculate the breakage rate by making a selection in the **PDF** drop-down list. You can select **parabolic**, **laakkonen**, **generalized**, or **user-defined** :

- If you select **parabolic**, the **Shape Factor for Parabolic PDF** dialog box will open automatically to enable you to specify the shape factor **C** (see [Figure 3.6: The Shape Factor for Parabolic PDF Dialog Box \(p. 430\)](#)). The PDF used in the breakage rate will then be calculated according to [Equation 2.20 \(p. 408\)](#) (as described in [Particle Birth and Death Due to Breakage and Aggregation \(p. 404\)](#)).

**Figure 3.6: The Shape Factor for Parabolic PDF Dialog Box**

- If you select **generalized**, the **Generalized pdf for multiple breakage** dialog box will open automatically ([Figure 3.7: The Generalized pdf for multiple breakage Dialog Box \(p. 430\)](#)).

**Figure 3.7: The Generalized pdf for multiple breakage Dialog Box**

Perform the following steps in the **Generalized pdf for multiple breakage** dialog box:

- Select either **One Term** or **Two Term** from the **Options** list. Your selection will determine whether  $i$  in [Equation 2.26 \(p. 409\)](#) is 0 or 1, respectively.
- Enter a value for the averaged **Number of Daughters**. It can be any real number (including non-integers, such as 2.5), as long as it is not less than 2.
- Define the parameter(s) for [Equation 2.26 \(p. 409\)](#) in the **Input Parameters** group box. When **One Term** is selected from the **Options** list, you must enter a value for **qi0**. When **Two Term** is selected from the **Options** list, you must enter values for **wi0**, **pi0**, **qi0**, **ri0**, and **qi1**. For information about

appropriate values for these parameters to result in the daughter distributions shown in [Table 2.3: Daughter Distributions \(p. 409\)](#), see [Table 2.4: Daughter Distributions \(cont.\) \(p. 409\)](#).

---

### Note

For the equal-size generalized pdf breakage distribution, the value for **qi0** should be set to 1e20.

---

- d. Click the **Validate/Apply** button to save the settings. The text boxes in the **All Parameters** group box will be updated, using the values you entered in the **Input Parameters** group box, as well as values derived from the constraints shown in [Equation 2.27 \(p. 409\) – Equation 2.29 \(p. 409\)](#).
- e. Verify that the values in the **All Parameters** group box represent your intended PDF before clicking **Close**.
  - If you have a user-defined function (UDF) that you want to use to model the PDF for the breakage rate, you can choose the **user-defined** option and specify the appropriate UDF. See [UDFs for Population Balance Modeling \(p. 447\)](#) for details about UDFs for the population balance model.

Choose between the default **Hagesather** formulation and the **Ramakrishna** formulation. Detailed information about these two methods can be found in [Breakage Formulations for the Discrete Method \(p. 416\)](#).

7. Enable **Include Expansion** if you want to account for bubble expansion due to large changes in hydrostatic pressure.
- 

### Note

The secondary phase must be modeled as compressible. This option is currently available for Discrete and QMOM only.

---

8. Specify the boundary conditions for the solution variables.

 **Setup** →  **Boundary Conditions**

See [Defining Population Balance Boundary Conditions \(p. 435\)](#) below.

9. Specify the initial guess for the solution variables.

 **Setup** →  **Solution Initialization**

10. Solve the problem and perform relevant postprocessing functions.

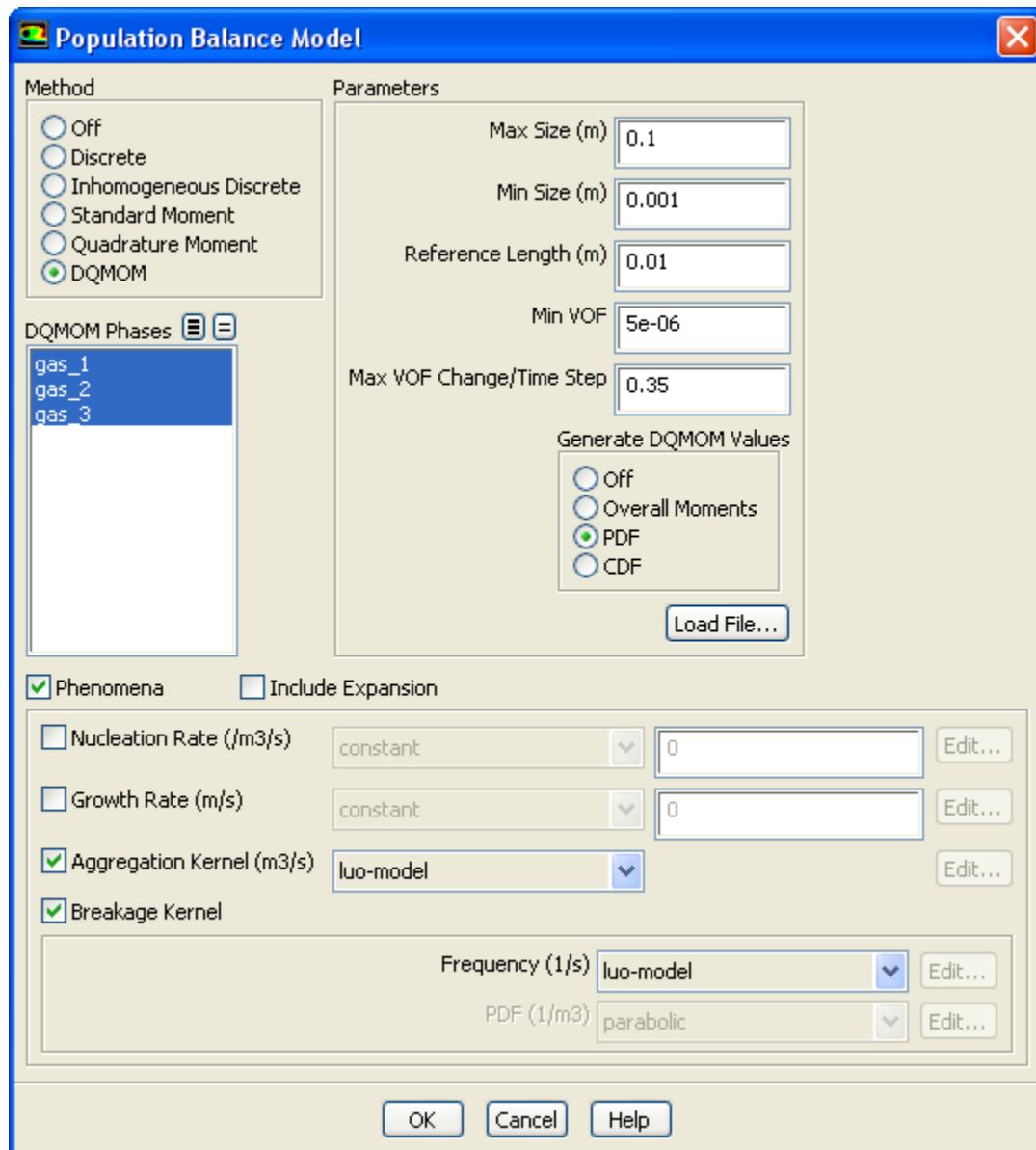
 **Setup** →  **Run Calculation**

See [Postprocessing for the Population Balance Model \(p. 443\)](#) for details about postprocessing.

### 3.3.1.1. Generated DQMOM Values

When using the **DQMOM** model, you have the option of generating DQMOM values from three different file formats. To do so, select **Overall Moments**, **PDF** or **CDF** under **Generate DQMOM Values** and then click the **Load File...** button. The **Select File** dialog box will open where you will select the appropriate file. Clicking **OK** in the **Select File** dialog box will result in the file being read and calculations carried out. The DQMOM values will be printed in the console.

**Figure 3.8: The Population Balance Model Dialog Box for the DQMOM Model**



#### DQMOM Values Produced From PDF, CDF Files, or Overall Moments for the Particles

- Three quadrature points are assumed, namely QP0, QP1, and QP2 (see [Figure 3.9: DQMOM Values Produced From a PDF File \(p. 433\)](#)).
- Length, Volume Fraction, and DQMOM-m4 values are given. The latter two can be used for initial fields of VOF and DQMOM as well as boundary conditions.

- For verification purposes, the first six moments are also given together with the total volume fraction of all particles from the PDF or CDF file. It is your responsibility to make sure that these values are correct, especially the total volume fraction.
- The definition of the first six moments of the particles is length based for the overall moments, described in [The Quadrature Method of Moments \(QMOM\) \(p. 418\)](#).
- In PDF or CDF format, the resultant volume fraction is normally given as unity. If you want the real particle volume fraction to be reflected in the mixture, the second column of the PDF or CDF data file need to be multiplied by the value of the real volume fraction. Another way is to multiply the values of the generated DQMOM volume fraction and DQMOM-m4 using the real particle volume fraction.

### Figure 3.9: DQMOM Values Produced From a PDF File

#### First 6 Moments computed from PDF:

```
m0 = 1.730757e+13
m1 = 5.442254e+08
m2 = 2.802300e+04
m3 = 1.909819e+00 (Total Volume Fraction is 9.999800e-01)
m4 = 1.533388e-04
m5 = 1.374813e-08
```

#### DQMOM values produced from PDF:

	QP0	QP1	QP2
Length (m)	1.050580e-04	5.154987e-05	1.282842e-05
Volume Fraction	5.452821e-01	4.433921e-01	1.130576e-02
DQMOM-m4 (m)	5.728627e-05	2.285681e-05	1.450350e-07

### PDF File Format

- The probability density function (PDF) is defined by the probability distribution of particles in terms of the volume fraction over the particle length (namely the diameter of the particle).
- The integration of PDF over all possible particle length (normally from 0 to the maximum diameter) shall give a value of unity or the real value of the volume fraction of all participating particles.
- The following file format is required (as shown below):
  - An integer number specified in the first line, indicating the number of data pairs to follow
  - The data in the 1st column specifying the length or diameter of particles in ascending order in meters (m)
  - The data in the 2nd column specifying the probability density function. Be aware that the integration of the PDF over the length shall result in a value of volume fraction for that particular particle length range

37  
 5e-6 0.000058e6  
 10e-6 0.000271e6  
 15e-6 0.000669e6  
 20e-6 0.001264e6  
 25e-6 0.002062e6  
 30e-6 0.003055e6  
 35e-6 0.004228e6  
 40e-6 0.005549e6  
 45e-6 0.006972e6

```
50e-6 0.008439e6
55e-6 0.00988e6
60e-6 0.011217e6
65e-6 0.012366e6
70e-6 0.013253e6
75e-6 0.013811e6
80e-6 0.013996e6
85e-6 0.013789e6
90e-6 0.013199e6
95e-6 0.012268e6
100e-6 0.011062e6
105e-6 0.009668e6
110e-6 0.00818e6
115e-6 0.006694e6
120e-6 0.00529e6
125e-6 0.004033e6
130e-6 0.002962e6
135e-6 0.002093e6
140e-6 0.00142e6
145e-6 0.000925e6
150e-6 0.000576e6
155e-6 0.000344e6
160e-6 0.000196e6
170e-6 0.000055e6
180e-6 0.000012e6
190e-6 0.000002e6
200e-6 0.
210e-6 0.
```

### CDF File Format

- The cumulative density function (CDF) is defined as the integration of PDF over all possible particles up to the length  $L$ , resulting in a value of volume fraction for all particles less than length  $L$ .
- The value of the CDF at the maximum particle length/diameter shall be unity, or the real value of the volume fraction of all particles in the mixture.
- The following file format is required (as shown below):
  - An integer number specified in the first line, indicating the number of data pairs to follow
  - The data in the 1st column specifying the length or diameter of particles in ascending order in meters (m)
  - The data in the 2nd column specifying the cumulative density function in terms of the volume fraction of particles
  - $CDF \leq 1$  at the maximum particle length

```
37
5e-6 0
10e-6 0.109e-2
15e-6 0.16e-2
20e-6 0.175e-2
25e-6 0.208e-2
30e-6 0.304e-2
35e-6 0.614e-2
40e-6 1.5e-2
44e-6 3.e-2
45e-6 3.44e-2
50e-6 6.573e-2
55e-6 11.19e-2
60e-6 17.11e-2
65e-6 24.17e-2
70e-6 32.007e-2
75e-6 40.243e-2
```

```

80e-6 48.397e-2
85e-6 56.453e-2
90e-6 63.823e-2
95e-6 70.53e-2
100e-6 76.073e-2
105e-6 81e-2
110e-6 85.057e-2
115e-6 88.187e-2
120e-6 90.89e-2
125e-6 92.897e-2
130e-6 94.437e-2
135e-6 95.543e-2
140e-6 96.523e-2
145e-6 97.173e-2
150e-6 97.717e-2
155e-6 98.007e-2
160e-6 98.363e-2
170e-6 98.88e-2
180e-6 99.16e-2
190e-6 99.327e-2
200e-6 100.e-2

```

### Overall Moments File Format

- The third option is to specify the first six moments for all particles as shown below.
- The following file format is required (as shown below):
  - An integer number specified in the first line, indicating the number of data (moments) to follow. By default, this shall be 6
  - six moments from moment-0 to moment-5 are given in that order. The definition of the first six moments is based on length as described in [The Quadrature Method of Moments \(QMOM\) \(p. 418\)](#)
  - As a check for moments,  $Moment-3(m3) = \frac{1}{(K_v)(volume\_fraction)}$ . In the values given below, volume fraction is assumed to be unity and  $K_v$  is assumed to be  $\pi/6$

```

6
1.120556e+013
4.022475e+008
2.523370e+004
1.909857e+000
1.611191e-004
1.498663e-008

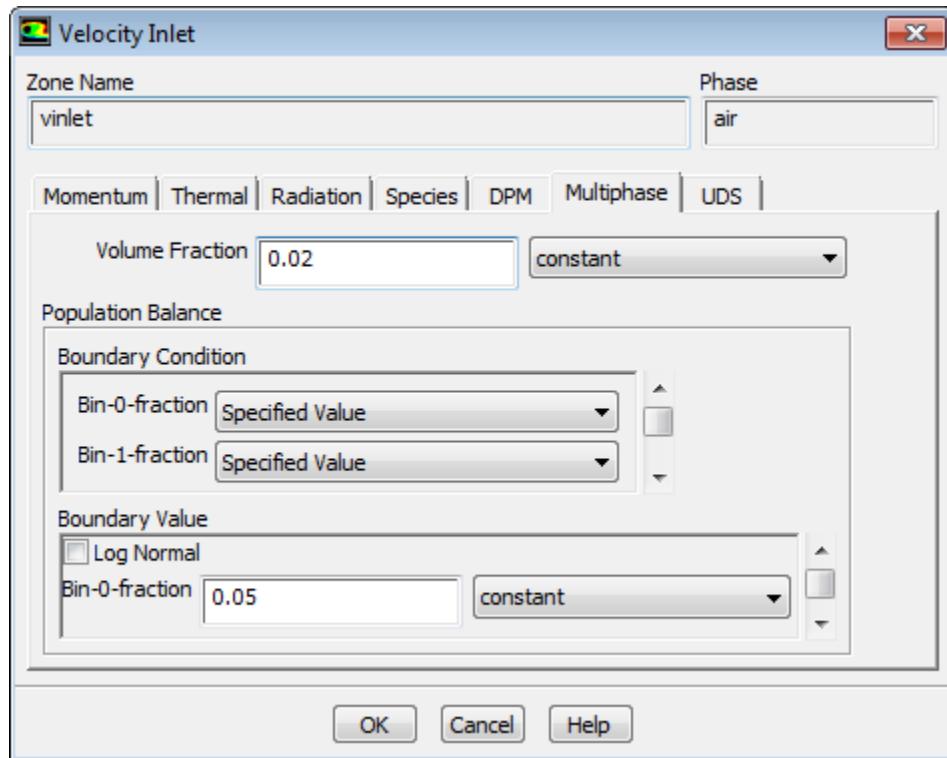
```

### 3.3.2. Defining Population Balance Boundary Conditions

To define boundary conditions specific to the population balance model, use the following procedure:

- In the **Boundary Conditions** task page, select the secondary phase(s) in the **Phase** drop-down list and then open the appropriate boundary condition dialog box (for example, [Figure 3.10: Specifying Inlet Boundary Conditions for the Population Balance Model \(p. 436\)](#)).

 **Setup** →  **Boundary Conditions**

**Figure 3.10: Specifying Inlet Boundary Conditions for the Population Balance Model**

2. In the **Multiphase** tab, under **Boundary Condition**, select the type of boundary condition for each bin (for the discrete method) or moment (for SMM and QMOM) as either **Specified Value** or **Specified Flux**.

Note that the boundary condition variables (for example, **Bin-0**) are labeled according to the following:

*bin/moment* - *i*th bin/moment

where the *i*th *bin/moment* can range from **0** (the first bin or moment) to *N*-1, where *N* is the number of bins/moments that you entered in the **Population Balance Model** dialog box.

3. Under **Population Balance Boundary Value**, enter a value or a flux as appropriate.

- If you selected **Specified Value** for the selected boundary variable, enter a value in the field adjacent to the variable name. This value will correspond to the variable  $f_i$  in [Equation 2.49 \(p. 415\)](#) (for the discrete method) or  $m_k$  in [Equation 2.64 \(p. 417\)](#) (for SMM or QMOM). If you are using either of the discrete methods and have selected **Specified Value** for all bins, you can optionally specify a log-normal distribution and have Fluent automatically initialize the bin fractions accordingly to a log-normal distribution (see [Initializing Bin Fractions With a Log-Normal Distribution \(p. 436\)](#)).
- If you selected **Specified Flux** for the selected boundary variable, enter a value in the field adjacent to the variable name. This value will be the spatial particle volume flux  $dV / dx_i$ .

### 3.3.2.1. Initializing Bin Fractions With a Log-Normal Distribution

When using one of the discrete methods with Specified Value selected as the boundary condition for all bins, you can have Fluent populate the bin fractions according to a log-normal distribution charac-

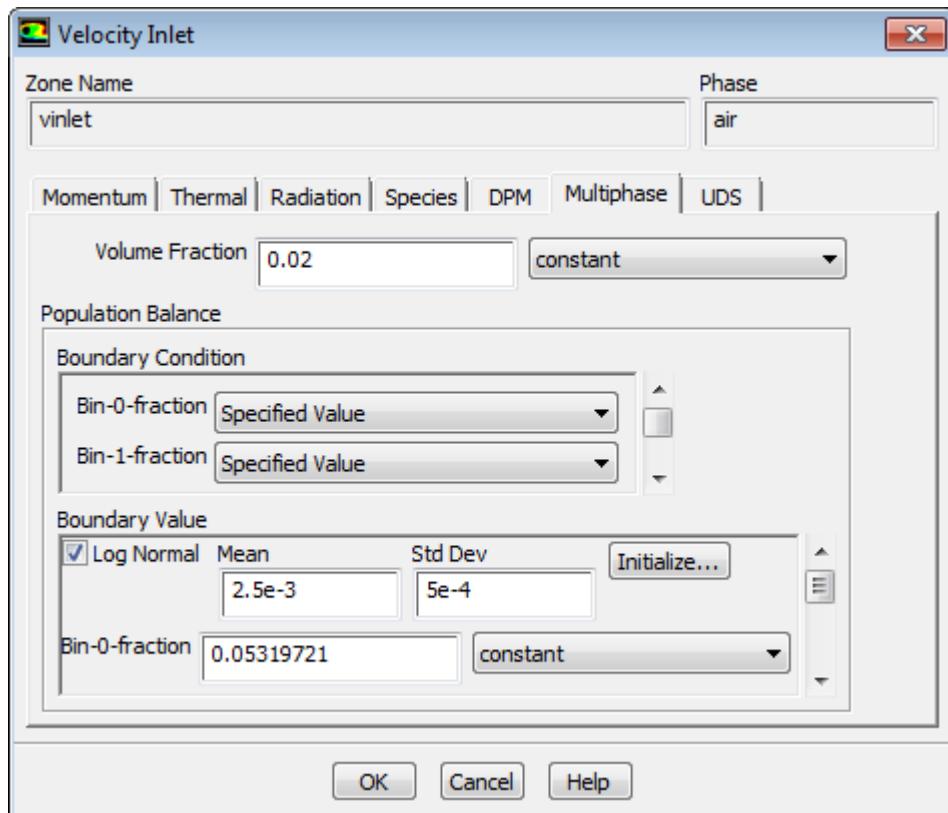
terized by a mean and standard deviation that you supply. For details of the log-normal distribution, refer to [The Log-Normal Distribution \(p. 422\)](#).

### Important

The log-normal initialization feature is only meaningful and should only be used when **Specified Value** is selected for all bins under **Boundary Condition**.

To use the log-normal distribution, perform the following steps.

1. Enable **Log Normal** in the **Boundary Value** group box.



2. Enter values for the **Mean** and **Std Dev** of the desired distribution.
3. Click **Initialize...**

### 3.3.3. Specifying Population Balance Solution Controls

In the **Equations** dialog box ([Figure 3.11: The Equations Dialog Box \(p. 438\)](#)), equations for each bin (for example, **phase-2 Bin**) will appear in the **Equations** list.

**Solution** → **Solution Controls** **Equations...**

The default value under **Under-Relaxation Factors** (in the **Solution Controls** task page) for the population balance equations is 0.5, and the default **Discretization** scheme (in the **Solution Methods** task page) is **First Order Upwind**.

**Figure 3.11: The Equations Dialog Box**

### 3.3.4. Coupling With Fluid Dynamics

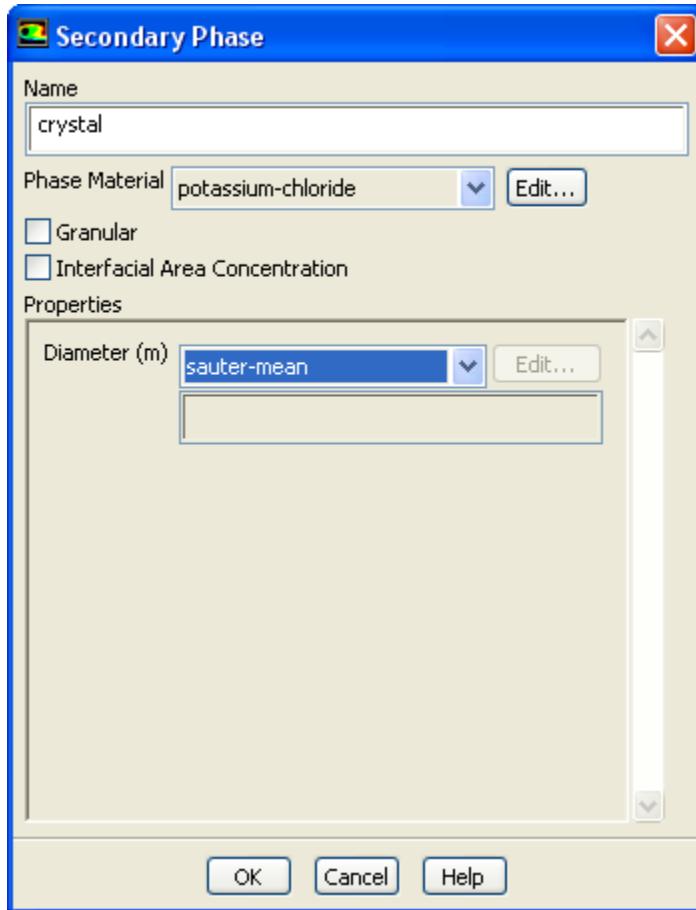
To couple population balance modeling of the secondary phase(s) with the overall problem fluid dynamics, a Sauter mean diameter ( $d_{32}$  in [Equation 2.71 \(p. 417\)](#)) may be used to represent the particle diameter of the secondary phase. The Sauter mean diameter is defined as the ratio of the third moment to the second moment for the SMM and QMOM. For the discrete method, it is defined as

$$d_{32} = \frac{\sum N_i L_i^3}{\sum N_i L_i^2} \quad (3.1)$$

To specify the Sauter mean diameter as the secondary phase particle diameter, open the **Secondary Phase** dialog box.

**Setup** → **Models** → **Multiphase** → **Phases** → **phase-id – Secondary Phase** **Edit...**

In the **Secondary Phase** dialog box (for example, [Figure 3.12: The Secondary Phase Dialog box for Hydrodynamic Coupling \(p. 439\)](#)), select **sauter-mean** from the **Diameter** drop-down list under **Properties**. Note that a constant diameter or user-defined function may also be used.

**Figure 3.12: The Secondary Phase Dialog box for Hydrodynamic Coupling**

### 3.3.5. Specifying Interphase Mass Transfer Due to Nucleation and Growth

In applications that involve the creation, dissolution, or growth of particles (such as crystallization), the total volume fraction equation for the particulate phase will have source terms due to these phenomena. The momentum equation for the particulate phase will also have source terms due to the added mass. In ANSYS Fluent, the mass source term can be specified using the UDF hook `DEFINE_HET_RXN_RATE`, as described in [Appendix A \(p. 455\)](#), or using the **Phase Interaction** dialog box, described below.

As an example, in crystallization, particles are created by means of nucleation ( $\dot{n}_0$ ), and a growth rate ( $G$ ) can also be specified. The mass transfer rate of formation (in  $\text{kg}/\text{m}^3\text{-s}$ ) of particles of all sizes is then

$$\begin{aligned}\dot{m} &= 3\rho K_v \int_0^\infty L^2 G n(L) dL \\ &= \frac{1}{2} \rho K_a \int_0^\infty L^2 G n(L) dL\end{aligned}\tag{3.2}$$

For the discrete method, the mass transfer rate due to growth can be written as

$$\begin{aligned}
 \dot{m} &= \rho \int_0^{\infty} G_v n(L) dL \\
 &= \rho \int_0^{\infty} G_v n(V) dV \\
 &= \rho \sum_i G_{v,i} N_i
 \end{aligned} \tag{3.3}$$

If the nucleation rate is included in the total mass transfer, then the mass transfer becomes

$$\dot{m} = \rho V_0 \dot{n}_0 + \sum_i \rho G_{v,i} N_i \tag{3.4}$$

### Important

For the discrete method, the sources to the population balance equations must sum to the total mass transfer rate. To access the sources, you can use the macro `C_PB_DISCI_PS` (`cell, thread, i`).

See [UDFs for Population Balance Modeling \(p. 447\)](#) for more information about macros for population balance variables.

For the SMM, only a size-independent growth rate is available. Hence, the mass transfer rate can be written as

$$\dot{m} = \frac{1}{2} \rho K_a G m \tag{3.5}$$

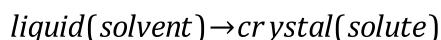
For the QMOM, the mass transfer rate can be written as

$$\dot{m} = \frac{1}{2} \rho K_a \sum_i L_i^2 w_i G(L_i) \tag{3.6}$$

For both the SMM and QMOM, mass transfer due to nucleation is negligible, and is not taken into account.

### Important

Note that for crystallization, the primary phase has multiple components; at the very least, there is a solute and a solvent. To define the multicomponent multiphase system, you will need to activate **Species Transport** in the **Species Model** dialog box for the primary phase after activating the multiphase model. The rest of the procedure for setting up a species transport problem is identical to setting up species in single phase. The heterogeneous reaction is defined as:



When the population balance model is activated, mass transfer between phases for non-reacting species (such as boiling) and heterogeneous reactions (such as crystallization) can be done automatically, in lieu of hooking a UDF.

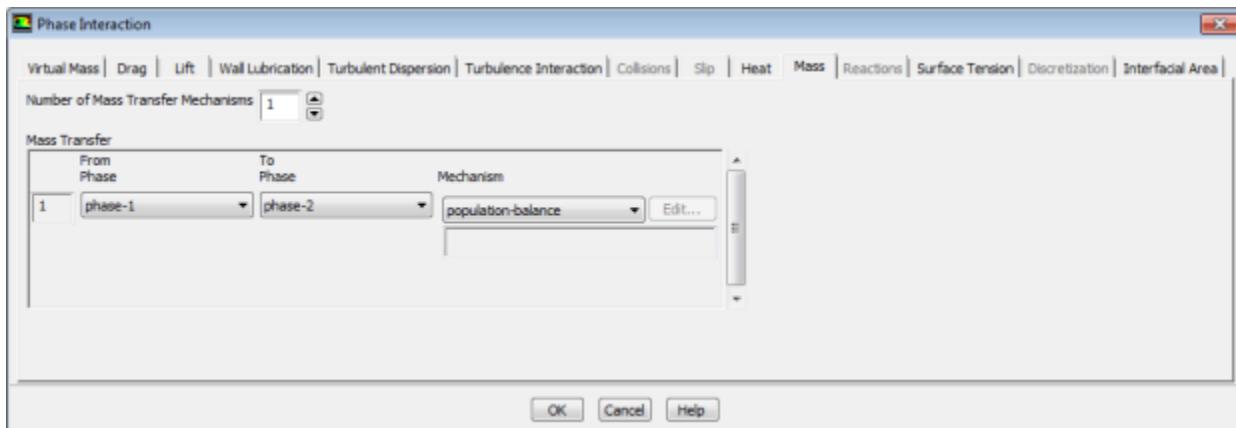
For simple unidirectional mass transfer between primary and secondary phases due to nucleation and growth phenomena of non-reacting species, configure the following settings:

1. Open the **Phase Interaction** dialog box (Figure 3.13: The Phase Interaction Dialog Box for Non-reacting Species (p. 441)).

**Setup** → **Models** → **Multiphase** → **Phases** → **Phase-Interactions** **Edit...**

2. Specify the mass transfer of species between the phases:

**Figure 3.13: The Phase Interaction Dialog Box for Non-reacting Species**



- a. Under the **Mass** tab, specify the **Number of Mass Transfer Mechanisms** involved in your case.
- b. For each mechanism, specify the phase of the source material under **From Phase** and the phase of the destination material phase under **To Phase**.
- c. From the **Mechanism** drop down list, select one of the mechanisms:

**none**

if you do not want any mass transfer between the phases.

**constant-rate**

for a fixed, user-specified rate.

**user-defined**

if you hooked a UDF describing the mass transfer mechanism.

**population-balance**

for an automated method of mass transfer, not involving a UDF. The nucleation and the growth rates calculated by the population balance kernels are used for mass transfer.

---

**Note**

For the **Inhomogeneous Discrete** population balance model, where there is more than one secondary phase, you can select population-balance as the mechanism of mass transfer between the solvent phase (say for crystallization) and each of the solute phases defined under the **Inhomogeneous Discrete** population balance model.

---

- d. Click **OK** to save the settings.

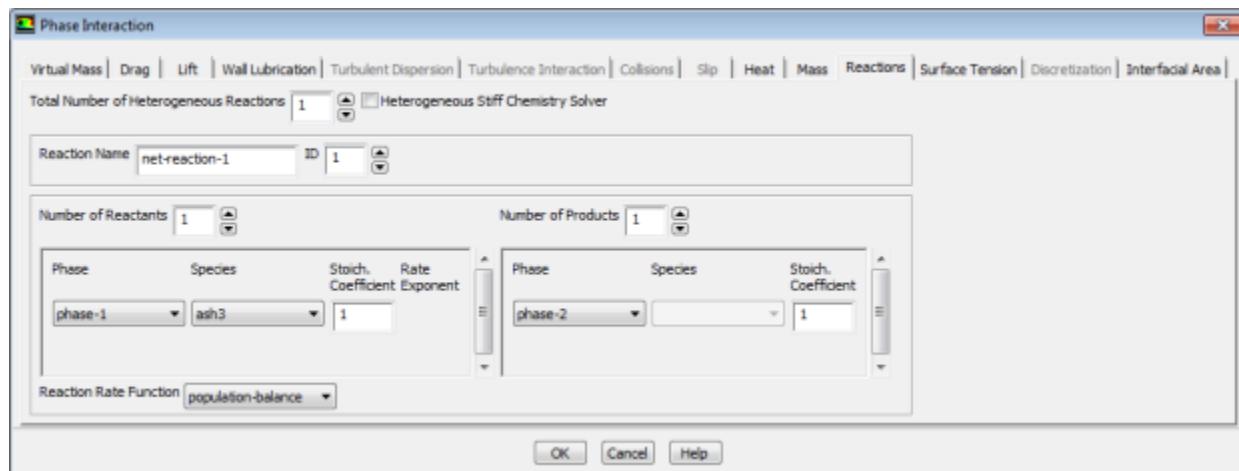
For heterogeneous reactions, configure the following settings:

1. For the primary phase, activate the **Species Transport** model.
2. Open the **Phase Interaction** dialog box (Figure 3.14: The Phase Interaction Dialog Box for a Heterogeneous Reaction (p. 442)).

**Setup** → **Models** → **Multiphase** → **Phases** → **Phase-Interactions** **Edit...**

3. Under the **Reactions** tab, specify the stoichiometry for the reactant and the product.

**Figure 3.14: The Phase Interaction Dialog Box for a Heterogeneous Reaction**



4. Select **population-balance** as the **Reaction Rate Function**.
5. Click **OK** to save the settings.

Either this method or the use of the UDF, described in [Appendix A \(p. 455\)](#), will produce the same results.

For the **Inhomogeneous Discrete** population balance model involving nucleation and growth, you can select **population-balance** as the **Reaction Rate Function** for each heterogeneous reaction you have set up. To learn how to set up reactions, go to [Specifying Heterogeneous Reactions](#) in the [User's Guide](#).

---

## Chapter 4: Postprocessing for the Population Balance Model

---

ANSYS Fluent provides postprocessing options for displaying, plotting, and reporting various particle quantities, which include the main solution variables and other auxiliary quantities.

- 4.1. Population Balance Solution Variables
- 4.2. Reporting Derived Population Balance Variables

### 4.1. Population Balance Solution Variables

Solution variables that can be reported for the population balance model are:

- **Bin-i fraction** (discrete method only), where i is N-1 bins/moments.
- **Number density of Bin-i fraction** (discrete method only)
- **Diffusion Coef. of Bin-i fraction/Moment-i**
- **Sources of Bin-i fraction/Moment-i**
- **Moment-i** (SMM and QMOM only)
- **Abscissa-i** (QMOM method only)
- **Weight-i** (QMOM method only)

**Bin-i fraction** is the fraction ( $f_i$ ) of the volume fraction for the  $i^{\text{th}}$  size bin when using the discrete method. **Number density of Bin-i fraction** is the number density ( $N_i$ ) in particles/m<sup>3</sup> for the  $i^{\text{th}}$  size bin. **Moment-i** is the  $i^{\text{th}}$  moment of the distribution when using the standard method of moments or the quadrature method of moments.

---

#### Important

Though the diffusion coefficients of the population variables (for example, **Diffusion Coef. of Bin-i fraction/Moment-i**) are available, they are set to zero because the diffusion term is not present in the population balance equations.

---

### 4.2. Reporting Derived Population Balance Variables

Two options are available in the **Postprocessing** ribbon tab that allow you to report computed moments and number density on selected surfaces or cell zones of the domain.

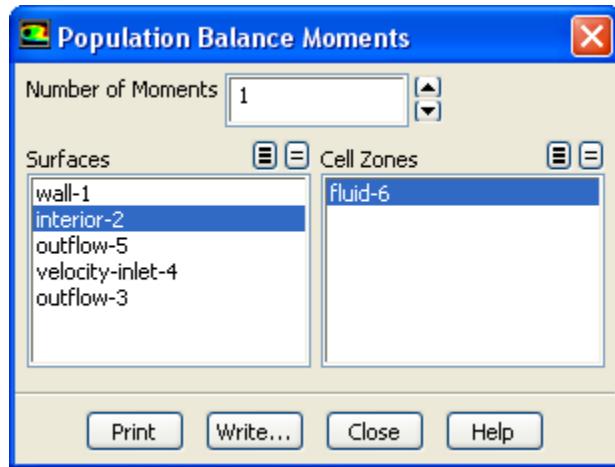
- 4.2.1. Computing Moments
- 4.2.2. Displaying a Number Density Function

## 4.2.1. Computing Moments

You can compute moments for the population balance model using the **Population Balance Moments** dialog box (Figure 4.1: The Population Balance Moments Dialog Box (p. 444)).

 Postprocessing → Model Specific → Population Balance → Moments...

**Figure 4.1: The Population Balance Moments Dialog Box**



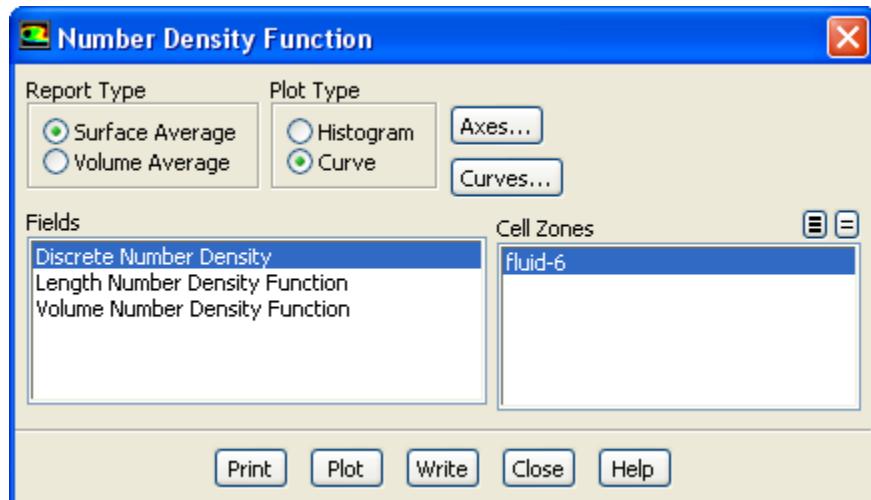
The steps for computing moments are as follows:

1. For the discrete method, specify the **Number of Moments**. For the SMM and QMOM, the number of moments is set equal to the number of moments that were solved, and therefore cannot be changed.
2. For a surface average, select the surface(s) on which to calculate the moments in the **Surfaces** list.
3. For a volume average, select the volume(s) in which to calculate the moments in the **Cell Zones** list.
4. Click **Print** to display the moment values in the console window.
5. To save the moment calculations to a file, click **Write...** and enter the appropriate information in the resulting **Select File** dialog box. The file extension should be .pb.

## 4.2.2. Displaying a Number Density Function

You can display the number density function for the population balance model using the **Number Density Function** dialog box (Figure 4.2: The Number Density Function Dialog Box (p. 445)).

 Postprocessing → Model Specific → Population Balance → Number Density...

**Figure 4.2: The Number Density Function Dialog Box**

The steps for displaying the number density function are as follows:

1. Specify the **Report Type** as either a **Surface Average** or a **Volume Average**.
2. Under **Plot Type**, specify how you would like to display the number density function data.

#### **Histogram**

displays a histogram of the discrete number density ( $N_i$ ). The number of divisions in the histogram is equal to the number of bins specified in the **Population Balance Model** dialog box. This option is available only with the discrete method.

#### **Curve**

displays a smooth curve of the number density function.

3. In the **Fields** list, select the data to be plotted.

#### **Discrete Number Density**

( $N_i$ ) is the number of particles per unit volume of physical space in the  $i^{\text{th}}$  size bin plotted against particle diameter size  $i$ . This option is available only with the discrete method.

#### **Length Number Density Function**

( $n(L)$ ) is the number of particles per unit volume of physical space per unit particle length plotted against particle diameter.

#### **Volume Number Density Function**

( $n(V)$ ) is the number of particles per unit volume of physical space per unit particle volume plotted against particle volume.

4. Choose the cell zones on which to plot the number density function data in the **Cell Zones** list.
5. Click **Plot...** to display the data.
6. (optional) Click **Print** to display the number density function data in the console window.
7. Click **Write** to save the number density function data to a file. The **Select File** dialog box will open, where you can specify a name and save a file containing the plot data. The file extension should be .pbd.



---

## Chapter 5: UDFs for Population Balance Modeling

---

This chapter contains the following sections:

- 5.1. Population Balance Variables
- 5.2. Population Balance DEFINE Macros
- 5.3. Hooking a Population Balance UDF to ANSYS Fluent

### 5.1. Population Balance Variables

The macros listed in [Table 5.1: Macros for Population Balance Variables Defined in sg\\_pb.h \(p. 447\)](#) can be used to return `real` variables associated with the population balance model. The variables are available in both the pressure-based and density-based solvers. The macros are defined in the `sg_pb.h` header file, which is included in `udf.h`.

**Table 5.1: Macros for Population Balance Variables Defined in sg\_pb.h**

Macro	Argument Types	Returns
C_PB_DISCI	<code>cell_t c, Thread *t, int i</code>	fraction ( $f_i$ ) of the total volume fraction for the $i^{\text{th}}$ size bin
C_PB_SMMI	<code>cell_t c, Thread *t, int i</code>	$i^{\text{th}}$ moment
C_PB_QMOMI	<code>cell_t c, Thread *t, int i</code>	$i^{\text{th}}$ moment, where $i=0,1,2,3,4,5$
C_PB_QMOMI_L	<code>cell_t c, Thread *t, int i</code>	abscissa $L_i$ , where $i=0,1,2$
C_PB_QMOMI_W	<code>cell_t c, Thread *t, int i</code>	weight $w_i$ , where $i=0,1,2$
C_PB_DISCI_PS	<code>cell_t c, Thread *t, int i</code>	net source term to $i^{\text{th}}$ size bin
C_PB_SMMI_PS	<code>cell_t c, Thread *t, int i</code>	net source term to $i^{\text{th}}$ moment
C_PB_QMOMI_PS	<code>cell_t c, Thread *t, int i</code>	net source term to $i^{\text{th}}$ moment

### 5.2. Population Balance DEFINE Macros

This section contains descriptions of `DEFINE` macros for the population balance model. Definitions of each `DEFINE` macro are contained in the `udf.h` header file.

- 5.2.1. `DEFINE_PB_BREAK_UP_RATE_FREQ`
- 5.2.2. `DEFINE_PB_BREAK_UP_RATE_PDF`
- 5.2.3. `DEFINE_PB_COALESCENCE_RATE`
- 5.2.4. `DEFINE_PB_NUCLEATION_RATE`
- 5.2.5. `DEFINE_PB_GROWTH_RATE`

## 5.2.1. DEFINE\_PB\_BREAK\_UP\_RATE\_FREQ

You can use the DEFINE\_PB\_BREAK\_UP\_RATE\_FREQ macro if you want to define the breakage frequency using a UDF. The function is executed at the beginning of every time step.

### 5.2.1.1. Usage

```
DEFINE_PB_BREAK_UP_RATE_FREQ(name, cell, thread, d_1)
```

Argument Type	Description
char name	UDF name
cell_t cell	Cell index
Thread *thread	Pointer to the secondary phase thread associated with d_1
real d_1	Parent particle diameter or length

#### Function returns

```
real
```

There are four arguments to DEFINE\_PB\_BREAK\_UP\_RATE\_FREQ: name, cell, thread, and d\_1. You will supply name, the name of the UDF. cell, thread, and d\_1 are variables that are passed by the ANSYS Fluent solver to your UDF.

### 5.2.1.2. Example

Included below is an example of a UDF that defines a breakage frequency (see [Particle Birth and Death Due to Breakage and Aggregation \(p. 404\)](#)) that is based on the work of Tavlarides [4] (p. 459), such that

$$g(V') = C_1 \frac{\varepsilon^{1/3}}{(1+\alpha)d^{2/3}} \exp\left(-C_2 \frac{\sigma(1+\alpha)^2}{\rho_l \varepsilon^{2/3} d^{5/3}}\right) \quad (5.1)$$

where  $C_1$  and  $C_2$  are constants,  $\varepsilon$  is the dissipation rate,  $d$  is the parent diameter,  $\sigma$  is the surface tension,  $\alpha$  is the volume fraction of the dispersed phase, and  $\rho_l$  is the density of the primary phase.

```
*****
UDF that computes the particle breakage frequency
*****
```

```
#include "udf.h"
#include "sg_pb.h"
#include "sg_mphase.h"

DEFINE_PB_BREAK_UP_RATE_FREQ(break_up_freq_tav, cell, thread, d_1)
{
    real epsi, alpha, f1, f2, rho_d;
    real C1 = 0.00481, C2 = 0.08, sigma = 0.07;
    Thread *tm = THREAD_SUPER_THREAD(thread); /*passed thread is phase*/
    epsi = C_D(cell, tm);
    alpha = C_VOF(cell, thread);
    rho_d = C_R(cell, thread);
    f1 = pow(epsi, 1./3.) / ((1.+epsi)*pow(d_1, 2./3.));
    f2 = -(C2*sigma*(1.+alpha)*(1.+alpha)) / (rho_d*pow(epsi, 2./3.)*pow(d_1, 5./3.));
    return C1*f1*exp(f2);
}
```

## 5.2.2. **DEFINE\_PB\_BREAK\_UP\_RATE\_PDF**

You can use the **DEFINE\_PB\_BREAK\_UP\_RATE\_PDF** macro if you want to define the breakage PDF using a UDF. The function is executed at the beginning of every time step.

### 5.2.2.1. **Usage**

```
DEFINE_PB_BREAK_UP_RATE_PDF(name, cell, thread, d_1, thread_2, d_2)
```

<b>Argument Type</b>	<b>Description</b>
char name	UDF name
cell_t cell	Cell index
Thread *thread	Pointer to the secondary phase thread associated with d_1
real d_1	Parent particle diameter or length
Thread *thread_2	Pointer to the secondary phase thread associated with d_2
real d_2	Diameter of one of the daughter particles after breakage; the second daughter particle diameter is calculated by conservation of particle volume

#### **Function returns**

```
real
```

There are six arguments to **DEFINE\_PB\_BREAK\_UP\_RATE\_PDF**: `name`, `cell`, `thread`, `d_1`, `thread_2`, and `d_2`. You will supply `name`, the name of the UDF. `cell`, `thread`, `d_1`, `thread_2`, and `d_2` are variables that are passed by the ANSYS Fluent solver to your UDF.

---

#### **Note**

`thread` and `thread_2` are the same for the Discrete, QMOM and SMM models. They may be the same or different depending on whether `d_1` and `d_2` belong to the same phase or different phases for the Inhomogeneous model.

---

### 5.2.2.2. **Example**

Included below is an example of a UDF that defines a breakage PDF (see [Particle Birth and Death Due to Breakage and Aggregation \(p. 404\)](#)) that is parabolic, as defined in [Equation 2.20 \(p. 408\)](#).

```
*****
UDF that computes the particle breakage PDF
*****
```

```
#include "udf.h"
#include "sg_pb.h"
#include "sg_mphase.h"

DEFINE_PB_BREAK_UP_RATE_PDF(break_up_pdf_par, cell, thread, d_1, thread_2, d_2)
{
    real pdf;
    real kv = M_PI/6.;

    real C = 1.0;

    real f_2, f_3, f_4;
    real V_prime = kv*pow(d_1,3.);
    real V = kv*pow(d_2,3.);
```

```
f_2 = 24.*pow(V/V_prime,2.);
f_3 = -24.*(V/V_prime);
f_4 = 6.;

pdf = (C/V_prime) + ((1.-C/2.)/V_prime)*(f_2 + f_3 + f_4);

return 0.5*pdf;
}
```

### 5.2.3. DEFINE\_PB\_COALESCENCE\_RATE

You can use the `DEFINE_PB_COALESCENCE_RATE` macro if you want to define your own particle aggregation kernel. The function is executed at the beginning of every time step.

#### 5.2.3.1. Usage

```
DEFINE_PB_COALESCENCE_RATE(name, cell, thread, d_1, thread_2, d_2)
```

Argument Type	Description
char name	UDF name
cell_t cell	Cell index
Thread *thread	Pointer to the secondary phase thread associated with d_1
Thread *thread_2	Pointer to the secondary phase thread associated with d_2
real d_1, d_2	Diameters of the two colliding particles

#### Function returns

```
real
```

---

There are six arguments to `DEFINE_PB_COALESCENCE_RATE`: `name`, `cell`, `thread`, `d_1`, `thread_2`, and `d_2`. You will supply `name`, the name of the UDF. `cell`, `thread`, `d_1`, and `d_2` are variables that are passed by the ANSYS Fluent solver to your UDF. Your UDF will need to return the `real` value of the aggregation rate.

---

#### Note

`thread` and `thread_2` are the same for the Discrete, QMOM and SMM models. They may be the same or different depending on whether `d_1` and `d_2` belong to the same phase or different phases for the Inhomogeneous model.

---

#### 5.2.3.2. Example

Included below is an example UDF for a Brownian aggregation kernel. In this example, the aggregation rate is defined as

$$a(L,\lambda) = a(V,V') = \beta_0 \frac{(L+\lambda)^2}{L\lambda}$$

where  $\beta_0 = 1 \times 10^{-17} \text{ m}^3/\text{s}$ .

```
*****
UDF that computes the particle aggregation rate
*****  
#include "udf.h"
```

```
#include "sg_pb.h"
#include "sg_mphase.h"

DEFINE_PB_COALESCENCE_RATE(aggregation_kernel,cell,thread,d_1,thread_2,d_2)
{
    real agg_kernel;
    real beta_0 = 1.0e-17 /* aggregation rate constant */
    agg_kernel = beta_0*pow((d_1+d_2),2.0)/(d_1*d_2);
    return agg_kernel;
}
```

## 5.2.4. DEFINE\_PB\_NUCLEATION\_RATE

You can use the `DEFINE_PB_NUCLEATION_RATE` macro if you want to define your own particle nucleation rate. The function is executed at the beginning of every time step.

### 5.2.4.1. Usage

```
DEFINE_PB_NUCLEATION_RATE(name, cell, thread)
```

Argument Type	Description
char name	UDF name
cell_t cell	Cell index
Thread *thread	Pointer to the secondary phase thread

#### Function returns

real

There are three arguments to `DEFINE_PB_NUCLEATION_RATE`: `name`, `cell`, and `thread`. You will supply `name`, the name of the UDF. `cell` and `thread` are variables that are passed by the ANSYS Fluent solver to your UDF. Your UDF will need to return the `real` value of the nucleation rate.

### 5.2.4.2. Example

Potassium chloride can be crystallized from water by cooling. Its solubility decreases linearly with temperature. Assuming power-law kinetics for the nucleation rate,

$$\dot{n}_n = K_n (S-1)^{N_n}$$

where  $K_n=4\times10^{10}$  particles/m<sup>3</sup>-s and  $N_n=2.77$ .

```
*****
UDF that computes the particle nucleation rate
*****
```

```
#include "udf.h"
#include "sg_pb.h"
#include "sg_mphase.h"
DEFINE_PB_NUCLEATION_RATE(nuc_rate, cell, thread)
{
    real J, S;
    real Kn = 4.0e10; /* nucleation rate constant */
    real Nn = 2.77; /* nucleation law power index */
    real T,solute_mass_frac,solvent_mass_frac, solute_mol_frac,solubility;
    real solute_mol_wt, solvent_mol_wt;

    Thread *tc = THREAD_SUPER_THREAD(thread); /*obtain mixture thread */
    Thread **pt = THREAD_SUB_THREADS(tc); /* pointer to sub_threads */
    Thread *tp = pt[P_PHASE]; /* primary phase thread */
```

```
solute_mol_wt = 74.55; /* molecular weight of potassium chloride */
solvent_mol_wt = 18.; /* molecular weight of water */
solute_mass_frac = C_YI(cell,tp,0);
/* mass fraction of solute in primary phase (solvent) */

solvent_mass_frac = 1.0 - solute_mass_frac;
solute_mol_frac = (solute_mass_frac/solute_mol_wt)/
((solute_mass_frac/solute_mol_wt)+(solvent_mass_frac/solvent_mol_wt));

T = C_T(cell,tp); /* Temperature of primary phase in Kelvin */

solubility = 0.0005*T-0.0794;
/* Solubility Law relating equilibrium solute mole fraction to Temperature*/

S = solute_mol_frac/solubility; /* Definition of Supersaturation */
if (S == 1.)
{
    J = 0.;
}
else
{
    J = Kn*pow((S-1),Nn);
}
return J;
```

---

## Important

Note that the solubility and the chemistry could be defined in a separate routine and simply called from the above function.

---

## 5.2.5. DEFINE\_PB\_GROWTH\_RATE

You can use the DEFINE\_PB\_GROWTH\_RATE macro if you want to define your own particle growth rate. The function is executed at the beginning of every time step.

### 5.2.5.1. Usage

```
DEFINE_PB_GROWTH_RATE(name, cell, thread,d_i)
```

Argument Type	Description
char name	UDF name
cell_t cell	Cell index
Thread *thread	Pointer to the secondary phase thread
real d_i	Particle diameter or length

### Function returns

```
real
```

---

There are four arguments to DEFINE\_PB\_GROWTH\_RATE: name, cell, thread, and d\_i. You will supply name, the name of the UDF. cell, thread, and d\_i are variables that are passed by the ANSYS Fluent solver to your UDF. Your UDF will need to return the real value of the growth rate.

### 5.2.5.2. Example

Potassium chloride can be crystallized from water by cooling. Its solubility decreases linearly with temperature. Assuming power-law kinetics for the growth rate,

$$G=K_g(S-1)^{N_g}$$

where  $K_g=2.8\times10^{-8}$  m/s and  $N_g=1$ .

```
*****
UDF that computes the particle growth rate
*****
```

```
#include "udf.h"
#include "sg_pb.h"
#include "sg_mphase.h"
DEFINE_PB_GROWTH_RATE(growth_rate, cell, thread,d_1)
{
    /* d_1 can be used if size-dependent growth is needed */
    /* When using SMM, only size-independent or linear growth is allowed */
    real G, S;
    real Kg = 2.8e-8; /* growth constant */
    real Ng = 1.; /* growth law power index */
    real T,solute_mass_frac,solvent_mass_frac, solute_mol_frac,solubility;
    real solute_mol_wt, solvent_mol_wt;

    Thread *tc = THREAD_SUPER_THREAD(thread); /*obtain mixture thread */
    Thread **pt = THREAD_SUB_THREADS(tc); /* pointer to sub_threads */
    Thread *tp = pt[P_PHASE]; /* primary phase thread */

    solute_mol_wt = 74.55; /* molecular weight of potassium chloride */
    solvent_mol_wt = 18.; /* molecular weight of water */
    solute_mass_frac = C_YI(cell, tp, 0);
    /* mass fraction of solute in primary phase (solvent) */

    solvent_mass_frac = 1.0 - solute_mass_frac;
    solute_mol_frac = (solute_mass_frac/solute_mol_wt)/
        ((solute_mass_frac/solute_mol_wt)+(solvent_mass_frac/solvent_mol_wt));

    T = C_T(cell, tp); /* Temperature of primary phase in Kelvin */
    solubility = 0.0005*T-0.0794;
    /* Solubility Law relating equilibrium solute mole fraction to Temperature*/

    S = solute_mol_frac/solubility; /* Definition of Supersaturation */
    if (S == 1.)
    {
        G = 0.;
    }
    else
    {
        G = Kg*pow((S-1),Ng);
    }
    return G;
}
```

## Important

Note that the solubility and the chemistry could be defined in a separate routine and simply called from the above function.

## 5.3. Hooking a Population Balance UDF to ANSYS Fluent

After the UDF that you have defined using `DEFINE_PB_BREAK_UP_RATE_FREQ`, `DEFINE_PB_BREAK_UP_RATE_PDF`, `DEFINE_PB_COALESCENCE_RATE`, `DEFINE_PB_NUCLEATION_RATE`, or `DEFINE_PB_GROWTH_RATE` is interpreted or compiled, the name that you specified in the `DEFINE` macro argument (for example, `agg_kernel`) will become visible and selectable in the appropriate drop-down list under **Phenomena** in the **Population Balance Model** dialog box (Figure 3.1: The Population Balance Model Dialog Box (p. 425)).



---

---

## Appendix A. DEFINE\_HET\_RXN\_RATE Macro

This appendix discusses the `DEFINE_HET_RXN_RATE` macro:

- A.1. Description
- A.2. Usage
- A.3. Example
- A.4. Hooking a Heterogeneous Reaction Rate UDF to ANSYS Fluent

### A.1. Description

You need to use `DEFINE_HET_RXN_RATE` to specify reaction rates for heterogeneous reactions. A heterogeneous reaction is one that involves reactants and products from more than one phase. Unlike `DEFINE_VR_RATE`, a `DEFINE_HET_RXN_RATE` UDF can be specified differently for different heterogeneous reactions.

During ANSYS Fluent execution, the `DEFINE_HET_RXN_RATE` UDF for each heterogeneous reaction that is defined is called in every fluid cell. ANSYS Fluent will use the reaction rate specified by the UDF to compute production/destruction of the species participating in the reaction, as well as heat and momentum transfer across phases due to the reaction.

A heterogeneous reaction is typically used to define reactions involving species of different phases. The bulk phase can participate in the reaction if the phase does not have any species (that is, the phase has fluid material instead of mixture material). Heterogeneous reactions are defined in the **Phase Interaction** dialog box.

### A.2. Usage

```
DEFINE_HET_RXN_RATE (name ,c ,t ,r ,mw ,yi ,rr ,rr_t)
```

Argument Type	Description
char name	UDF name
cell_t c	Cell index
Thread *t	Cell thread (mixture level) on which heterogeneous reaction rate is to be applied
Hetero_Reaction *r	Pointer to data structure that represents the current heterogeneous reaction (see <code>sg_mphase.h</code> )
real mw[MAX_PHASES][MAX_SPE_EQNS]	Matrix of species molecular weights. <code>mw[i][j]</code> will give molecular weight of species with ID <code>j</code> in phase with index <code>i</code> . For phase that has fluid material, the molecular weight can be accessed as <code>mw[i][0]</code> .
real yi[MAX_PHASES][MAX_SPE_EQNS]	Matrix of species mass fractions. <code>yi[i][j]</code> will give molecular weight of species with ID <code>j</code> in phase with index <code>i</code> . For phase that has fluid material, <code>yi[i][0]</code> will be 1.

```
real *rr           Pointer to laminar reaction rate
real *rr_t         Currently not used. Provided for future use.

Function returns
void
```

There are eight arguments to DEFINE\_HET\_RXN\_RATE: name, c, t, r, mw, yi, rr, and rr\_t. You will supply name, the name of the UDF. c, t, r, mw, yi, rr, and rr\_t are variables that are passed by the ANSYS Fluent solver to your UDF. Your UDF will need to set the values referenced by the real pointer rr.

## A.3. Example

The following compiled UDF, named arrh, defines an Arrhenius-type reaction rate. The rate exponents are assumed to be same as the stoichiometric coefficients.

```
#include "udf.h"

static const real Arrhenius = 1.e15;
static const real E_Activation = 1.e6;
#define SMALL_S 1.e-29

DEFINE_HET_RXN_RATE(arrh,c,t,hr,mw,yi,rr,rr_t)
{
    Domain **domain_reactant = hr->domain_reactant;
    real *stoich_reactant = hr->stoich_reactant;
    int *reactant = hr->reactant;
    int i;
    int sp_id;
    int dindex;
    Thread *t_reactant;
    real ci;
    real T = 1200.; /* should obtain from cell */

    /* instead of compute rr directly, compute log(rr) and then take exp */

    *rr = 0;
    for (i=0; i < hr->n_reactants; i++)
    {
        sp_id = reactant[i]; /* species ID to access mw and yi */
        if (sp_id == -1) sp_id = 0; /* if phase does not have species,
                                       mw, etc. will be stored at index 0 */
        dindex = DOMAIN_INDEX(domain_reactant[i]);
        /* domain index to access mw & yi */
        t_reactant = THREAD_SUB_THREAD(t,dindex);

        /* get conc. */
        ci = yi[dindex][sp_id]*C_R(c,t_reactant)/mw[dindex][sp_id];
        ci = MAX(ci,SMALL_S);
        *rr += stoich_reactant[i]*log(ci);
    }

    *rr += log(Arrhenius + SMALL_S) -
           E_Activation/(UNIVERSAL_GAS_CONSTANT*T);

    /* 1.e-40 < rr < 1.e40 */
    *rr = MAX(*rr,-40);
    *rr = MIN(*rr,40);

    *rr = exp(*rr);
}
```

## A.4. Hooking a Heterogeneous Reaction Rate UDF to ANSYS Fluent

After the UDF that you have defined using `DEFINE_HET_RXN_RATE` is interpreted or compiled (see the [Fluent Customization Manual](#) for details), the name that you specified in the `DEFINE` macro argument (for example, `arrh`) will become visible and selectable under **Reaction Rate Function** in the **Reactions** tab of the **Phase Interaction** dialog box. (Note you will first need to specify the **Total Number of Reactions** greater than **0**.)



# Bibliography

- [1] J. Abrahamson. "Collision Rates of Small Particles in a Vigorously Turbulent Fluid". *Chemical Engineering Science*. 30. 1371–1379. 1975.
- [2] L. Austin, K. Shoji, V. Bhatia, V. Jindal, K. Savage, and R. Kliment. "Some Results on the Description of Size Distribution as a Rate Process in Various Mills". *Industrial Engineering and Chemical Process Design Devices*. 15(1). 187–196. 1976.
- [3] J. Baldyga and W. Orciuch. "Barium Sulfate Precipitation in a Pipe – An Experimental Study and CFD Modeling". *Chemical Engineering Science*. 56. 2435–2444. 2001.
- [4] C. A. Couloglou and L. L. Tavlarides. "Description of Interaction Processes in Agitated Liquid-Liquid Dispersions". *Chemical Engineering Science*. 32. 1289–1297. 1977.
- [5] H. Dette and W. J. Studden. *The Theory of Canonical Moments with Applications in Statistics, Probability, and Analysis*. John Wiley & Sons. New York, NY1997.
- [6] R. B. Diemer and J. H. Olson. "A Moment Methodology for Coagulation and Breakage Problems Part 3—Generalized Daughter Distribution Functions". *Chemical Engineering Science*. 57(19). 4187–4198. 2002.
- [7] R. Fan, D.L. Marchisio and R.O. Fox. *Application of the Direct Quadrature Method of Moments to Polydisperse Gas-solid Fluidized Beds*. *Power Technology*. 139. 7–20. 2004.
- [8] M. Ghadiri and Z. Zhang. "Impact Attrition of Particulate Solids Part 1. A Theoretical Model of Chipping". *Chemical Engineering Science*. 57. 3659–3669. 2002.
- [9] R. G. Gordon. "Error Bounds in Equilibrium Statistical Mechanics". *Journal of Mathematical Physics*. 56. 655–633. 1968.
- [10] Hagesaether L., Jakobsen H.A.1, and Svendsen H.F. "A Model for Turbulent Binary Breakup of Dispersed Fluid Particles". *Chemical Engineering Science*. 57(16). 3251–3267. 2002.
- [11] K. Higashitani, K. Yamauchi, Y. Matsuno, and G. Hosokawa. "Turbulent Coagulation of Particles Dispersed in a Viscous Fluid". *Chemical Engineering Journal Japan*. 16(4). 299–304. 1983.
- [12] M. J. Hounslow, R. L. Ryall, and V. R. Marshall. "A Discretized Population Balance for Nucleation, Growth, and Aggregation". *AIChE Journal*. 34(11). 1821–1832. 1988.
- [13] M. A. Hsia and L. L. Tavlarides. "Simulation analysis of drop breakage, coalescence and micro-mixing in liquid–liquid stirred tanks". *Chemical Engineering Journal*. 26(3). 189–199. 1983.
- [14] M. Kostoglou, S. Dovas, and A. J. Karabelas. "On the Steady-State Size Distribution of Dispersions in Breakage Processes". *Chemical Engineering Science*. 52(8). 1285–1299. 1997.
- [15] S. Kumar and D. Ramkrishna. "On the Solution of Population Balance Equations by Discretization - I., A Fixed Pivot Technique". *Chemical Engineering Science*. 51(8). 1311–1332. 1996.
- [16] F. Lehr, M. Millies, and D. Mewes. "Bubble-Size Distributions and Flow Fields in Bubble Columns". *AIChE Journal*. 8(11). 2426–2443. 2002.
- [17] J. D. Litster, D. J. Smit, and M. J. Hounslow. "Adjustable Discretization Population Balance for Growth and Aggregation". *AIChE Journal*. 41(3). 591–603. 1995.

## Bibliography

---

- [18] H. Luo. "Coalescence, Breakup and Liquid Circulation in Bubble Column Reactors". PhD thesis from the Norwegian Institute of Technology. Trondheim,Norway1993.
- [19] H. Luo and H. F. Svendsen. "Theoretical Model for Drop and Bubble Breakup in Turbulent Dispersions". *AIChE Journal*. 42(5). 1225–1233,. 1996.
- [20] D. L. Marchisio, R. D. Virgil, and R. O. Fox. "Quadrature Method of Moments for Aggregation-Breakage Processes". *Journal of Colloid and Interface Science*. 258. 322–334. 2003.
- [21] B. J. McCoy and M. Wang. "Continuous-mixture Fragmentation Kinetics Particle Size Reduction and Molecular Cracking". *Chemical Engineering Science*. 49(22). 3773–3785. 1994.
- [22] R. McGraw. "Description of Aerosol Dynamics by the Quadrature Method of Moments". *Aerosol Science and Technology*. 27. 255–265. 1997.
- [23] R. Moreno-Atanasio and M. Ghadiri. "Mechanistic Analysis and Computer Simulation of Impact Breakage of Agglomerates Effect of Surface Energy". *Chemical Engineering Science*. 61(8). 2476–2481. 2006.
- [24] S. B. Pope. "Probability Distributions of Scalars in Turbulent Shear Flow". *Turbulent Shear Flows*. volume 2. pages 7–16. 1979.
- [25] W. H. Press, S. A. Teukolsky, W.T. Vetterling, and B. P. Flannery. *Numerical Recipes*. Cambridge University Press. Cambridge, England,1992.
- [26] D. Ramkrishna. *Population Balances Theory and Applications to Particulate Systems in Engineering*. Academic Press. San Diego, CA,2000.
- [27] A. D. Randolph and M. A. Larson. *Theory of Particulate Processes. Analysis and Techniques of Continuous Crystallization*. Academic Press. San Diego, CA1971.
- [28] P.G. Saffman and J.S. Turner. *On the Collision of Droplets in Turbulent Clouds*. *Journal of Fluid Mechanics*,. 1. 16–30. 1956.
- [29] M. Schmoluchowski. *Versuch Einen Mathematischen Theorie der Koagulationstechnik Kolloider Lösungen*. *Zeitschrift für Physikalische Chemie*. 92. 129–168. 1917.
- [30] R. D.Vigil and R. M.Ziff. *On the Stability of Coagulation–Fragmentation Population Balances*. *Journal of Colloids and Interface Science*. 133(1). 257–264. 1989.