

**DECLARATIVE MODELING OF COUPLED ADVECTION AND DIFFUSION
AS APPLIED TO FUEL CELLS**

A Dissertation
Presented to
The Academic Faculty

by

Kevin L. Davies

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
George W. Woodruff School of Mechanical Engineering

Georgia Institute of Technology
May 2014

Copyright © 2014 by Kevin L. Davies

**DECLARATIVE MODELING OF COUPLED ADVECTION AND DIFFUSION
AS APPLIED TO FUEL CELLS**

Approved by:

Dr. Christiaan J.J. Paredis,
Committee Chair
George W. Woodruff School of Mechanical
Engineering
Georgia Institute of Technology

Dr. Sheldon M. Jeter
George W. Woodruff School of Mechanical
Engineering
Georgia Institute of Technology

Dr. Comas L. Haynes, Advisor
Georgia Tech Research Institute
Georgia Institute of Technology

Dr. Thomas F. Fuller
Chemical and Biological Engineering
Georgia Institute of Technology

Dr. Tequila A.L. Harris
George W. Woodruff School of Mechanical
Engineering
Georgia Institute of Technology

Dr. Robert M. Moore
Hawaii Natural Energy Institute
University of Hawaii

Date Approved: 8 January 2014

Copyright © 2014 by Kevin L. Davies
kdavies4@gmail.com



This work is licensed under a Creative Commons Attribution-ShareAlike 3.0 Unported License.
Source files for this document are available upon request.

To Everett, Colin, and Morgan

ACKNOWLEDGEMENTS

I would like to acknowledge the Georgia Tech Research Institute for the Robert G. Shackelford Fellowship and the George W. Woodruff School of Mechanical Engineering for the Presidential Fellowship.

I am grateful to my advisors, Dr. Chris Paredis and Dr. Comas Haynes, for their support and patience. I thank Dr. Robert Moore for providing the initial opportunity to pursue this line of research and for the wisdom and guidance he provided along the way. I thank the rest of my committee, Dr. Thomas Fuller, Dr. Tequila Harris, and Dr. Sheldon Jeter, for their careful review and valuable comments.

I appreciate the caring attitude and professionalism of the staff of the George W. Woodruff School of Mechanical Engineering. They have always been extremely helpful, positive, and understanding.

I am thankful for my labmates who have offered ideas, moral support, and friendship: Bill Bailey, Bill Binder, Douglas Broadwell, Chris Ford, Sebastian Herzig, Katie Hornbostel, Dimitri Hughes, Alisha Kasam, Alek Kerzhner, Ben Lee, Malik Little, Rich Malak, Roxanne Moore, George Nelson, Axel Reichwein, Muhammad Salman, Brian Taylor, and Stephanie Thompson. I appreciate the help of Kevin Bandy with data analysis.

I thank Mike Angelo and Guido Bender for providing the benchmark fuel cell data from the Hawaii Natural Energy Institute.

I would also like to acknowledge the open-source software community. Many free tools have been very helpful, including Modelica [1] and the Modelica Standard Library [2], Python, SciPy and NumPy [3], matplotlib [4], Wikipedia, Inkscape, LibreOffice, LaTeX, Kile, JabRef, git, Linux, and Ubuntu.

Finally, I thank my family. I have been blessed with a family who has given me confidence, encouragement, and endless support. They are so caring and empathetic that they have unfortunately felt the burden of this work with me. My dad patiently listened as I thought through many of the ideas in this dissertation.

TABLE OF CONTENTS

DEDICATION	iii
ACKNOWLEDGEMENTS	iv
LIST OF TABLES	xiii
LIST OF FIGURES	xiv
LISTS OF ABBREVIATIONS	xviii
NOMENCLATURE	xx
GLOSSARY	xxiii
SUMMARY	xxv
1 INTRODUCTION	1
1.1 Context and Motivation	2
1.1.1 Advantages of Declarative Modeling	3
1.1.2 Current Limitations	6
1.2 Research Questions	8
1.3 Application to Fuel Cells	8
1.3.1 Context and Motivation	8
1.3.2 Research Questions	15
1.4 Overview of the Modeling Approach	15
1.5 Outline of the Dissertation	17
2 BACKGROUND	19
2.1 Equation-Based, Object-Oriented (EEO) Modeling Languages	19
2.1.1 Modelica	20
2.1.2 VHDL-AMS	22
2.1.3 Verilog-AMS	22
2.1.4 gPROMS	22
2.1.5 Simscape	23
2.2 Fluid/Chemical Modeling in Modelica	23
2.2.1 Without Coupled Advection	24
2.2.2 Central Difference	24

2.2.3	Without Coupled Diffusion	25
2.3	Fuel Cell Models	29
2.3.1	Physics-Based	29
2.3.2	Phenomenological	33
2.3.3	Declarative	34
2.4	Summary	37
3	FUNDAMENTALS OF THE MODEL	39
3.1	Correlated Thermodynamic Properties	43
3.1.1	Isobaric Specific Heat Capacity-Temperature Relation	43
3.1.2	Pressure-Volume-Temperature Relation	44
3.2	Derived Thermodynamic Properties	45
3.2.1	Specific Entropy	46
3.2.2	Specific Enthalpy	47
3.2.3	Specific Gibbs Energy	48
3.2.4	Isobaric Specific Heat Capacity	48
3.2.5	Isochoric Specific Heat Capacity	48
3.3	Mixtures	49
3.3.1	Species within a Phase	49
3.3.2	Phases within a Region	50
3.4	Basic Conservation Equations	50
3.4.1	Material	51
3.4.2	Rotational Momentum	52
3.4.3	Translational Momentum	53
3.4.4	Energy	55
3.5	Exchange Equations	57
3.5.1	Phase Change	61
3.5.2	Drag and Translational Advection	64
3.5.3	Thermal Conduction and Advection	66
3.6	Exchange Properties	69
3.7	Transport Equations	70
3.7.1	Material Transport	84

3.7.2	Normal Translational Advection and Nonequilibrium Compression . . .	87
3.7.3	Transverse Translational Advection and Friction	90
3.7.4	Thermal Advection and Conduction	96
3.8	Transport Properties	99
3.9	Electrochemical Reactions	100
3.9.1	Context	100
3.9.2	Equations	102
3.10	Detailed Conservation Equations	108
3.10.1	Material	109
3.10.2	Translational Momentum	111
3.10.3	Energy	114
3.11	Summary	118
4	IMPLEMENTATION OF THE MODEL	119
4.1	Introduction	119
4.2	Quantities	121
4.3	Units	121
4.4	Characteristics	123
4.5	Connectors	124
4.6	Species	128
4.7	Chemistry	131
4.7.1	Reactions	131
4.7.2	Electron Transfer and Charge Storage	132
4.7.3	Capillary Pressure	133
4.8	Phases	133
4.9	Subregions	137
4.10	Regions/Layers	140
4.11	Assemblies/Cells	142
4.12	Test Stand	144
4.13	Summary	146
5	BASIC EXAMPLES	147
5.1	Internal Flow	148

5.2	Echo	153
5.3	Air Column	160
5.4	Electrical Conduction	165
5.5	Thermal Conduction	168
5.6	Thermal Conduction and Convection	170
5.7	Evaporation	173
5.8	Hydration	176
5.9	Summary	178
6	SIMULATION OF THE FUEL CELL MODEL	181
6.1	Baseline Polarization Test	181
6.2	Varying Temperature	198
6.3	Varying Pressure	199
6.4	Varying Anode Flow Rate	200
6.5	Varying Cathode Flow Conditions	201
6.6	Varying Anode Humidity	204
6.7	Varying Cathode Humidity	204
6.8	Simple Cell Model	206
6.9	Segmented Cell	208
6.10	Cyclical Load	214
6.11	Summary	219
7	CONCLUSIONS	220
7.1	Recapitulation	220
7.2	Contributions	225
	7.2.1 Primary	225
	7.2.2 Secondary	228
7.3	Future Work	229
7.4	Final Comments	230
	APPENDIX A RELATED THEORY	231
A.1	Darcy's Law	231
A.2	Maxwell-Stefan Equations	233
A.3	Charge Drift and Diffusion	239

A.4	Ohm's Law	240
A.5	Einstein Relation	241
APPENDIX B SELECTED DOCUMENTATION AND SOURCE CODE		243
B.1	FCSys.Assemblies.Cells.Examples.TestConditions	243
B.2	FCSys.Assemblies.Cells.Examples.TestStand	247
B.3	FCSys.Characteristics.BaseClasses.Characteristic	249
B.4	FCSys.Characteristics.BaseClasses.Characteristic.c_p	254
B.5	FCSys.Characteristics.BaseClasses.Characteristic.c_v	256
B.6	FCSys.Characteristics.BaseClasses.Characteristic.eta	257
B.7	FCSys.Characteristics.BaseClasses.Characteristic.h	258
B.8	FCSys.Characteristics.BaseClasses.Characteristic.mu	261
B.9	FCSys.Characteristics.BaseClasses.Characteristic.nu	262
B.10	FCSys.Characteristics.BaseClasses.Characteristic.s	263
B.11	FCSys.Characteristics.BaseClasses.Characteristic.tauprime	266
B.12	FCSys.Characteristics.BaseClasses.Characteristic.theta	268
B.13	FCSys.Characteristics.BaseClasses.Characteristic.zeta	269
B.14	FCSys.Characteristics.BaseClasses.CharacteristicEOS	270
B.15	FCSys.Characteristics.BaseClasses.CharacteristicEOS.p_Tv	272
B.16	FCSys.Characteristics.BaseClasses.CharacteristicEOS.v_Tp	273
B.17	FCSys.Characteristics.BaseClasses.CharacteristicNASA	274
B.18	FCSys.Characteristics.BaseClasses.CharacteristicNASA.eta	279
B.19	FCSys.Characteristics.BaseClasses.CharacteristicNASA.theta	280
B.20	FCSys.Characteristics.'C+'.Graphite	281
B.21	FCSys.Characteristics.'e-'.Graphite	281
B.22	FCSys.Characteristics.'H+'.Ionomer	282
B.23	FCSys.Characteristics.'SO3-'.Ionomer	282
B.24	FCSys.Characteristics.H2.Gas	283
B.25	FCSys.Characteristics.H2O.Gas	284
B.26	FCSys.Characteristics.H2O.Ionomer	284
B.27	FCSys.Characteristics.H2O.Liquid	284
B.28	FCSys.Characteristics.MobilityFactors.BaseClasses.k	285

B.29 FCSys.Characteristics.N2.Gas	288
B.30 FCSys.Characteristics.O2.Gas	288
B.31 FCSys.Chemistry.Capillary	289
B.32 FCSys.Chemistry.CapillaryVolume	290
B.33 FCSys.Chemistry.Electrochemistry.DoubleLayer	293
B.34 FCSys.Chemistry.Electrochemistry.ElectronTransfer	296
B.35 FCSys.Conditions.Adapters.ChemicalReaction	299
B.36 FCSys.Conditions.Environment	300
B.37 FCSys.Connectors	302
B.38 FCSys.Connectors.Amagat	307
B.39 FCSys.Connectors.Boundary	308
B.40 FCSys.Connectors.Chemical	308
B.41 FCSys.Connectors.Dalton	309
B.42 FCSys.Connectors.Inert	310
B.43 FCSys.Connectors.Reaction	311
B.44 FCSys.Connectors.ThermalDiffusive	312
B.45 FCSys.Connectors.Translational	313
B.46 FCSys.Phases.PartialPhase	313
B.47 FCSys.Quantities	316
B.48 FCSys.Regions.AnCLs.AnCL	320
B.49 FCSys.Regions.AnFPs.AnFP	322
B.50 FCSys.Regions.AnGDLs.AnGDL	324
B.51 FCSys.Regions.CaCLs.CaCL	326
B.52 FCSys.Regions.CaFPs.CaFP	328
B.53 FCSys.Regions.CaGDLs.CaGDL	330
B.54 FCSys.Regions.PEMs.PEM	332
B.55 FCSys.Regions.Region	333
B.56 FCSys.Species.'C+'.Graphite.Fixed	336
B.57 FCSys.Species.'e-'.Graphite.Fixed	337
B.58 FCSys.Species.'H+'.Ionomer.Fixed	338
B.59 FCSys.Species.'SO3 ⁻² '.Ionomer.Fixed	339

B.60 FCSys.Species.Enumerations	340
B.61 FCSys.Species.Fluid	342
B.62 FCSys.Species.H2.Gas.Fixed	359
B.63 FCSys.Species.H2O.Gas.Fixed	360
B.64 FCSys.Species.H2O.Ionomer.Fixed	361
B.65 FCSys.Species.H2O.Liquid.Fixed	362
B.66 FCSys.Species.Ion	363
B.67 FCSys.Species.N2.Gas.Fixed	364
B.68 FCSys.Species.O2.Gas.Fixed	365
B.69 FCSys.Species.Solid	366
B.70 FCSys.Species.Species	370
B.71 FCSys.Subregions.Subregion	379
B.72 FCSys.Units	382
B.73 FCSys.Units.Bases	391
B.74 FCSys.Units.Bases.Base	392
B.75 FCSys.Utilities	393
B.76 FCSys.Utilities.Coordinates	394
B.77 FCSys.Utilities.Polynomial	395
REFERENCES	396

LIST OF TABLES

2.1	Relative occurrence of declarative modeling languages	21
2.2	Features of selected physics-based fuel cell models	32
3.1	Cross-references of physical topics and laws	40
3.2	Limiting cases of the transport equation	76
3.3	Scenarios of one-dimensional advection with diffusion	81
3.4	Values of the translational Nusselt number	92
4.1	Effort/flow pairs of the connectors	126
5.1	Modeling and simulation statistics for the internal flow example	149
5.2	Modeling and simulation statistics for the echo example	153
6.1	Key parameters of the fuel cell model	183
6.2	Modeling and simulation statistics for the simple fuel cell model	206
6.3	Modeling and simulation statistics for the segmented fuel cell model	209
7.1	Effort/flow pairs of the connectors (duplicate of Table 4.1)	221

LIST OF FIGURES

1.1	Imperative and declarative models of an electrical circuit	5
1.2	Inverse imperative model of the circuit in Figure 1.1	6
1.3	Depiction of advection and diffusion	7
1.4	Layers of a single-cell proton exchange membrane fuel cell (PEMFC)	9
1.5	PEMFC used to power an unmanned aerial vehicle	9
1.6	PEMFC system used to power a bus	11
1.7	Configurations of two PEMFC systems	12
1.8	Levels of physical hierarchy in the model library	16
3.1	Illustration of a region or subregion	41
3.2	Considerations of a fuel cell model	42
3.3	Integration path for the specific entropy of gases	47
3.4	Types of material intake considered in the model	52
3.5	Types of forces considered in the model	54
3.6	Types of energy intake considered in the model	55
3.7	Locations of the phase change processes	62
3.8	Property in the bulk of a region due to advection and diffusion	77
3.9	Property at the interface between regions due to advection and diffusion	78
3.10	Profile of a property between regions due to advection and diffusion	79
3.11	Transport rate of a conserved quantity under mixed advection and diffusion	83
3.12	Weighting scheme to achieve zero torque	94
3.13	Thermal convection in the model	97
3.14	Location of the electrochemical reaction	101
3.15	Advection, diffusion, and properties along the reaction coordinate	103
4.1	Levels of instantiation in the model library (duplicate of Figure 1.8)	120
4.2	Parameter dialog for the H ₂ species	131
4.3	Diagrams of the reaction models	132
4.4	Diagrams of the phases	136
4.5	Parameter dialog for the gas phase	136
4.6	Diagram of a subregion	137

4.7	Parameter dialog for a subregion	139
4.8	Diagrams of a region	140
4.9	Parameter dialog for the anode flow plate	141
4.10	Single-cell PEMFC	143
4.11	Diagram of the fuel cell test stand	145
5.1	Configuration of the internal flow example	150
5.2	Fluid velocity for the internal flow example	151
5.3	Dynamic pressure difference under internal flow	152
5.4	Thermal transients due to viscous dissipation in internal flow	152
5.5	Configuration of the echo example (H ₂ gas with an initial pressure difference)	154
5.6	Pressure dynamics in the echo example	156
5.7	Temperature oscillations in the echo example	157
5.8	Long-term thermal equilibration in the echo example	158
5.9	Velocity in the echo example	159
5.10	Configuration of the air column example (gas under the influence of gravity)	160
5.11	Pressure in the air column example	162
5.12	Velocity transients in the air column example	163
5.13	Thermal transients in the air column example	164
5.14	Thermal time constants in the air column example	165
5.15	Configuration of the electrical conduction example	166
5.16	Dynamic heating of an electrical resistor	167
5.17	Configuration of the thermal conduction example	168
5.18	Temperature in a graphite bar during transient conduction	169
5.19	Configuration of the thermal conduction and convection example	170
5.20	Velocity induced in gas in contact with graphite undergoing thermal conduction	172
5.21	Pressure induced in gas in contact with graphite undergoing thermal conduction	172
5.22	Configuration of the evaporation and condensation example	173
5.23	Pressure of H ₂ O reaching saturation	174
5.24	Rate of evaporation of initially sub-saturated H ₂ O	175
5.25	Temperature of H ₂ O during evaporation	175
5.26	Validation of H ₂ O saturation pressure	176

5.27 Configuration of the hydration example	177
5.28 Hydration level of the ionomer reaching equilibrium	178
5.29 Rate of hydration of the ionomer	179
5.30 Equilibrium hydration level versus relative humidity	179
6.1 Polarization curves of the cell for the baseline polarization	185
6.2 Energy balance under the baseline conditions at 1.5 A/cm^2	186
6.3 Potential losses through the cell during the baseline polarization	187
6.4 Electrical losses in the cell during the baseline polarization	188
6.5 Temperatures through the cell during the baseline polarization	188
6.6 Total pressure of the gas during the baseline polarization	189
6.7 Pressure drop down the flow channels during the baseline polarization	190
6.8 Velocity of H_2O down the channels during the baseline polarization	190
6.9 H_2 pressure during the baseline polarization	191
6.10 O_2 pressure during the baseline polarization	191
6.11 H_2O pressure during the baseline polarization	192
6.12 H_2O balance under the baseline conditions at 1.5 A/cm^2	193
6.13 H_2O transport during the baseline polarization	193
6.14 H_2O transport through the membrane during the baseline polarization	194
6.15 Hydration during the baseline polarization test	195
6.16 Liquid pore saturation during the baseline polarization	195
6.17 Relative humidity through the cell during the baseline polarization	196
6.18 Rates of evaporation during the baseline polarization	197
6.19 Rates of hydration during the baseline polarization	197
6.20 Polarization curves with varying temperature	199
6.21 Polarization curves with varying outlet pressure	200
6.22 Polarization curves with varying anode flow rates	201
6.23 Polarization curves with varying cathode flow rates and compositions	203
6.24 Polarization curves with varying humidity at the anode inlet	204
6.25 Polarization curves with varying humidity at the cathode inlet	205
6.26 Polarization curves for the standard and simplified cell models	207
6.27 Relative humidity throughout the simplified cell model	208

6.28	Net polarization of the segmented cell under stoichiometric and fixed flow	210
6.29	Pressure down the channels of the segmented cell	211
6.30	Pressure of O ₂ in the cathode catalyst layer of the segmented cell	212
6.31	Temperature in the cathode catalyst layer of the segmented cell	212
6.32	Overpotentials in the cathode of the cell segments	213
6.33	Rates of heat generation due to the ORR in the cell segments	213
6.34	Model diagram for the test with a cyclical load	214
6.35	Cyclical load applied to the cell (sinusoidal, reversing)	215
6.36	Current and voltage under cyclical load	216
6.37	Temperatures of throughout the cell under cyclical load	217
6.38	Gas pressure throughout the cell under cyclical load	217
6.39	Relative humidity under cyclical load	218
6.40	Liquid pore saturation under cyclical load	218
6.41	Hydration under cyclical load	219

LISTS OF ABBREVIATIONS

Standard

- an. Anode
ca. Cathode

Acronyms

0D	<u>0</u> -dimensional
1D	<u>1</u> -dimensional
2D	<u>2</u> -dimensional
3D	<u>3</u> -dimensional
ASIC	<u>A</u> pplication-specific <u>i</u> ntegrated <u>c</u> ircuit
BOP	<u>B</u> alance <u>o</u> f <u>p</u> lant
CDF	<u>C</u> omputable <u>D</u> ocument <u>F</u> ormat
CFD	<u>C</u> omputational <u>f</u> luid <u>d</u> ynamics
DAE	<u>D</u> ifferential <u>a</u> lgebraic <u>e</u> quation
DASSL	<u>D</u> ifferential/ <u>A</u> lgebraic <u>S</u> ystem <u>S</u> olver <u>L</u> ibrary
EDL	<u>E</u> lectrolytic <u>d</u> ouble <u>l</u> ayer
EOO	<u>E</u> quation-based, <u>o</u> bject- <u>o</u> riented
EOS	<u>E</u> quation <u>o</u> f <u>s</u> tate
FC	<u>F</u> uel <u>c</u> ell
FDM	<u>F</u> inite <u>d</u> ifference <u>m</u> ethod
FEM	<u>F</u> inite <u>e</u> lement <u>m</u> ethod
FHM	<u>F</u> unctional <u>h</u> ybrid <u>m</u> odeling
FMI	<u>F</u> unctional <u>m</u> ock-up <u>i</u> nterface
FVM	<u>F</u> inite <u>v</u> olume <u>m</u> ethod
GDL	<u>G</u> as diffusion layer
HDL	<u>H</u> ardware <u>d</u> escription <u>l</u> anguage
HNEI	<u>H</u> awaii <u>N</u> atural <u>E</u> nergy <u>I</u> nstitute
HOR	<u>H</u> ydrogen <u>o</u> xidation <u>r</u> eaction ($\text{H}_2 \rightarrow 2\text{H}^+ + 2\text{e}^-$)
IC	<u>I</u> nitial <u>c</u> ondition
ICE	<u>I</u> nternal <u>c</u> ombustion <u>e</u> ngine
ISO	<u>I</u> nternational <u>S</u> tandards <u>O</u> rganization
KCL	<u>K</u> irchhoff's <u>c</u> urrent <u>l</u> aw [5]
KVL	<u>K</u> irchhoff's <u>v</u> oltage <u>l</u> aw [5]
MEA	<u>M</u> embrane <u>e</u> lectrode <u>a</u> ssembly
MOL	<u>M</u> ethod <u>o</u> f <u>l</u> ines
MOR	<u>M</u> odel <u>o</u> rders <u>r</u> eduction
MPC	<u>M</u> odel <u>p</u> redictive <u>c</u> ontrol
ODE	<u>O</u> rdinary <u>d</u> ifferential <u>e</u> quation
ORR	<u>O</u> xygen <u>r</u> eduction <u>r</u> eaction ($4\text{H}^+ + 4\text{e}^- + \text{O}_2 \rightarrow 2\text{H}_2\text{O}$)
PDE	<u>P</u> artial <u>d</u> ifferential <u>e</u> quation
PEM	<u>P</u> roton <u>e</u> xchange (or <u>p</u> olymer <u>e</u> lectrolyte) <u>m</u> embrane
PEMFC	<u>P</u> roton <u>e</u> xchange (or <u>p</u> olymer <u>e</u> lectrolyte) <u>m</u> embrane <u>f</u> uel <u>c</u> ell
PEN	<u>P</u> ositive <u>e</u> lectrolyte <u>n</u> egative
SI	<u>I</u> nternational System of Units (<u>S</u> ystème <u>i</u> nternational d'unités)

SOFC	Solid oxide fuel cell
SSSF	Steady state, steady flow (noun or adjective)
UUV	Unmanned underwater vehicle

Electrochemical Species

C^+	Carbon ions
$C_{19}HF_{37}O_5S^-$	Nafion sulfonate ions
e^-	Electrons
H^+	Protons
H_2	Diatomic hydrogen
H_2O	Water
N_2	Diatomic nitrogen
O_2	Diatomic oxygen
Pt^+	Platinum ions
SO_3^-	Sulfonic acid

NOMENCLATURE^{i ii}

Symbols

<i>A</i>	Area [L^2]
<i>a</i>	Thermodynamic activity [1] or acceleration due to body forces [LT^{-2}]
<i>b</i>	Correlation coefficient [misc.]
<i>C</i>	Heat capacity [N]
<i>c</i>	Specific heat capacity ($c \equiv C/N$) [1]
<i>D</i>	Hydraulic diameter [L] or self diffusivity [$L^2 T^{-1}$]
<i>d</i>	Specific particle diameter [LN^{-1}]
<i>E</i>	Electric field [$LMN^{-1} T^{-2}$]
<i>f</i>	Force [$LM T^{-2}$]
<i>G</i>	Generalized conductance [misc.]
<i>g</i>	Specific Gibbs energy ($g \equiv G/N$) [$L^2 MN^{-1} T^{-2}$]
<i>H</i>	Enthalpy [$L^2 M T^{-2}$]
<i>h</i>	Specific enthalpy ($h \equiv H/N$) [$L^2 MN^{-1} T^{-2}$] or heat transfer coefficient [$NL^{-2} T^{-1}$]
<i>I</i>	Material current [NT^{-1}]
<i>i</i>	Integer index [-]
<i>J</i>	Generic flux [misc. $\times L^{-2}$]
<i>j</i>	Integer index [-]
<i>k</i>	Dimensionless constant or correlation coefficient [1]
<i>Kn</i>	Knudsen number [1]
<i>L</i>	Length [L]
<i>M</i>	Mass [M]
<i>m</i>	Specific mass ($m \equiv M/N$) [MN^{-1}] or cardinality [-]
<i>mρ</i>	Density [ML^{-3}]
<i>mΦ</i>	Translational momentum [$LM T^{-1}$]
<i>N</i>	Particle number (i.e., amount of material) [N]
<i>n</i>	Misc. ratio (e.g., stoichiometric coefficient) [1] or integer [-]
<i>Nu</i>	Nusselt number [1]
<i>P</i>	Perimeter [L]
<i>p</i>	Thermodynamic pressure [$ML^{-1} T^{-2}$]
<i>Pe</i>	Péclet number [1]
<i>Q</i>	Heat (valid only in the context of transfer, e.g., \dot{Q}) [$L^2 MT^{-2}$]
<i>q</i>	Single particle [N]
<i>R</i>	Generalized resistance [misc.] or electrical resistance [$ML^2 T^{-3}$]
<i>r</i>	Generalized resistivity [misc.]
<i>Re</i>	Reynolds number [1]
<i>S</i>	Entropy [N]
<i>s</i>	Specific entropy ($s \equiv S/N$) [1]
<i>T</i>	Temperature [$L^2 MN^{-1} T^{-2}$]
<i>t</i>	Time [T]

ⁱThe dimensions are noted in square brackets in terms of angle (A), length (L), mass (M), particle number (N), and time (T). Please see Section 4.2 for details.

ⁱⁱAlthough not shown here, boldface is used to denote vector quantities and operators.

T_s	Temperature-specific entropy product [$L^2 M N^{-1} T^{-2}$]
U	Internal energy [$L^2 M T^{-2}$]
V	Volume [L^3]
ν	Specific volume ($\nu \equiv V/N$) [$L^3 N^{-1}$]
w	Electrical potential [$L^2 M N^{-1} T^{-2}$]
X	Generic transferred quantity [misc.]
x	Position along the x axis or generic axis of transport [L]
Z	Charge [N]
z	Charge number (i.e., specific charge; $z \equiv Z/N$) [1]
zI	Electrical current [$N T^{-1}$]
α	Adjusted collision interval [T] or charge transfer coefficient [1]
β	Dynamic compressibility [$N T L^{-2} M^{-1}$]
γ	Generic driving property for exchange or transport [misc.]
ε	Porosity [1] or permittivity [$N^2 T^2 L^{-3} M^{-1}$]
η	Fluidity [$L T M^{-1}$]
ζ	Self resistivity [$T L^{-2}$]
θ	Thermal resistivity [$L T N^{-1}$]
κ	Permeability (Darcy's law) [L^2]
λ	Mean free path [L]
μ	Mobility [$N T M^{-1}$] or dynamic viscosity [$M L^{-1} T^{-1}$]
ν	Thermal independency [T]
Ξ	Set of sets ξ
ξ	Set of configurations undergoing exchange through a node
ρ	Concentration (i.e., volumic amount) [$N L^{-3}$]
σ	Shear shape factor [1] or electrical conductivity [$N T L^{-3}$]
τ	Time constant [T] or mean collision interval [T]
Φ	Product of particle number and velocity [$L N T^{-1}$]
ϕ	Translational velocity [$L T^{-1}$]
χ	Mole fraction [1]
ω	Flow diversion angle [A]

Accents

· Flow rate of $_$ [$\times T^{-1}$]

Superscripts

^o Nominal $_$ or $_$ at reference state
 ' Modified, effective, or characteristic $_$
 * Dimensionless $_$ [\times misc.]

Subscripts

1, 2, 3, or ... First, second, third, or ... $_$
 A $_$ of advection or advective
 a, b, c, or ... First, second, third, or ... $_$
 c $_$ as, of, or in the condensed phase
 conf $_$ of configurations

D	_ of <u>d</u> iffusion or <u>d</u> iffusive
DOF	_ of <u>d</u> egrees of <u>f</u> reedom
E	_ of <u>e</u> xchange
e^-	_ of electrons
eq	_ at or of equilibrium
g	_ as, of, or in gas
gen	_ of generation
H	_ of monoatomic hydrogen
H^+	_ of protons
H_2	_ of hydrogen
H_2O	_ of water
i	_ as, of, or in ionomer
i	_ of index i
IG	_ of <u>i</u> deal <u>g</u> as
j	_ of index j
l	_ as, of, or in liquid
N	Material _
n	_ of index n
n	_ at or into the negative boundary
O_2	_ of oxygen
p	Isobaric _
p	_ at or into the positive boundary
phases	_ of phases
Q	Thermal _
spec	_ of species
T	Isothermal _
T	_ of <u>t</u> ransport
tot	Total _
v	Isochoric _
x	_ along the x axis (through the cell)
y	_ along the y axis (down the channel)
z	_ along the z axis (across the channel)
ξ	_ of set ξ
ζ	_ for fluidity
θ	_ for thermal resistivity
Φ	Translational _
\perp	_ perpendicular to the axis of transport
\parallel	_ parallel to the axis of transport

GLOSSARY

acausal ~ (<i>adj</i>) of a modeling formalism using equations (also declarative, equation-based, or physical-interaction)	3
advection ~ (<i>noun</i>) transfer of a quantity due to sustained transfer of material (also drift or migration in the context of material transport)	7
causal ~ (<i>adj</i>) of a modeling formalism using assignments (also imperative, sequential modular, or signal-flow)	3
configuration ~ (<i>noun</i>) a species in a certain phase (e.g., H ₂ O vapor or H ₂ O liquid) . . .	41
conversion property ~ (<i>noun</i>) an effective intensive property of the sources (i.e., reactants) in advective exchange [misc.]	59
conversion velocity ~ (<i>noun</i>) the mass-weighted average velocity of the sources in advective translational exchange [L T ⁻¹]	65
current ~ (<i>noun</i>) material flow rate [N T ⁻¹]	41
declarative ~ (<i>adj</i>) of a modeling formalism using equations (also acausal, equation-based, or physical-interaction)	3
diffusion ~ (<i>noun</i>) transfer of a quantity that leads to homogeneity due to collisions or thermal agitation of material, without sustained material transfer	7
dynamic compressibility ~ (<i>noun</i>) the extent to which a non-equilibrium normal force causes or requires transient compression [N T L ⁻² M ⁻¹]	89
effort ~ (<i>noun</i>) a property or a variable representing a property that is equal among branches of a node (also effort variable or across variable)	6
equivalent current ~ (<i>noun</i>) the rate of supply of a reactant required to support the given electrical current assuming the reactant is entirely consumed (complete utilization)	208
exchange ~ (<i>noun</i>) the local transfer of a conserved quantity among a set of configurations	57
flow ~	
1. (<i>noun</i>) a one-dimensional vector that sums to zero into a node (also flow variable or through variable)	6
2. (<i>noun</i>) a quantity that is transferred; may be used as a noun modifier as in flow rate	6
imperative ~ (<i>adj</i>) of a modeling formalism using assignments (also causal, sequential modular, or signal-flow)	3
independence ~ (<i>noun</i>) generalized resistance to diffusive exchange (e.g., mobility) . . .	68
independity ~ (<i>noun</i>) specific independence	68
irreversible advection ~ (<i>noun</i>) the effective transfer of a quantity due to material flow projected across the difference in a driving property between a transition and the source configuration or a boundary and the upstream subregion	82

material ~	
1. (<i>noun</i>) the quantity that represents particles, atoms, or molecules (i.e., matter)	41
2. (<i>adj</i>) of particles, atoms, or molecules	41
mediation property ~ (<i>noun</i>) a conductance-weighted average property of a set of configurations that interact within a subregion [misc.]	60
mediation temperature ~ (<i>noun</i>) the conductance-weighted average temperature of a set of configurations that interact by thermal diffusion within a subregion [$L^2 M N T^{-2}$]	68
mediation velocity ~ (<i>noun</i>) the conductance-weighted average velocity of a set of configurations that interact by translational diffusion within a subregion [$L T^{-1}$]	66
phase ~ (<i>noun</i>) a control volume that contains a species or mixture of species as a solid, liquid, or gas	41
region ~ (<i>noun</i>) a control volume that contains one or more subregions	41
species ~	
1. (<i>noun</i>) a substance with a particular chemical formula (e.g., C^+ , e^- , or O_2) without regards to the phase	41
2. (<i>noun</i>) in the model library, a control volume that contains material with a particular chemical formula and the associated momentum and energy	128
specific ~ an adjective that indicates the quotient of the following quantity and its associated particle number [$_ \times N^{-1}$]	42
state ~	
1. (<i>noun</i>) in a mathematical model, a scalar time-varying variable of which a derivative is taken	120
2. (<i>noun</i>) in thermodynamics, the condition of a system, which may encompass multiple properties	120
subregion ~ (<i>noun</i>) a control volume that contains phases	39
transfer ~ (<i>noun</i>) transport or exchange	41
translational Nusselt number ~ (<i>noun</i>) the correction factor in the shear force equation (3.127) or Newton's law of viscous shear for the shape of the flow profile (also shear shape factor) [1]	91
transport ~ (<i>noun</i>) the transfer of a conserved quantity from one location to another	71

SUMMARY

The goal of this research is to realize the advantages of declarative modeling for complex physical systems that involve both advection and diffusion to varying degrees in multiple domains. This occurs, for example, in chemical devices such as fuel cells. The declarative or equation-based modeling approach can provide computational advantages and is compatible with physics-based, object-oriented representations. However, there is no generally accepted method of representing coupled advection and diffusion in a declarative modeling framework.

This work develops, justifies, and implements a new upstream discretization scheme for mixed advective and diffusive flows that is well-suited for declarative models. The discretization scheme yields a gradual transition from pure diffusion to pure advection without switching events or nonlinear systems of equations. Transport equations are established in a manner that ensures the conservation of material, momentum, and energy at each interface and in each control volume. The approach is multi-dimensional and resolved down to the species level, with conservation equations for each species in each phase. The framework is applicable to solids, liquids, gases, and charged particles. Interactions among species are described as exchange processes which are diffusive if the interaction is inert or advective if it involves chemical reactions or phase change.

The equations are implemented in a highly modular and reconfigurable manner using the Modelica language. A wide range of examples are demonstrated—from basic models of electrical conduction and evaporation to a comprehensive model of a proton exchange membrane fuel cell (PEMFC). Several versions of the PEMFC model are simulated under various conditions including polarization tests and a cyclical electrical load. The model is shown to describe processes such as electro-osmotic drag and liquid pore saturation. It can be scaled in complexity from 4000 to 32,000 equations, resulting in a simulation times from 0.2 to 19 s depending on the level of detail. The most complex example is a seven-layer cell with six segments along the

length of the channel. The model library is thoroughly documented and made available as a free, open-source software package.

INTRODUCTION

This dissertation concerns the problem of representing coupled advection and diffusion in a manner that is physics-based, modular, reconfigurable, and leads to numerically efficient and robust models. In complex physical systems, advection and diffusion are coupled to varying degrees in multiple domains. This occurs to the extent that there is both (1) translation of material with respect to a reference frame or exchange of material between phases or chemical forms and (2) a gradient or species-to-species variation in an intensive property (e.g., temperature, density, or velocity) that tends to become uniform due to thermal activity. It is known that the declarative or equation-based modeling approach can provide computational advantages and is compatible with physics-based, object-oriented representations. However, there is no generally accepted method of representing coupled advection and diffusion in a declarative modeling framework.

In this dissertation, we will present an approach to this problem and apply it to fuel cells. Fuel cells exhibit many processes which involve advection and diffusion to varying degrees, including chemical reactions, phase change, electrical conduction, fluid transport, multicomponent diffusion, and heat transfer. In addition to being a pertinent demonstration platform, fuel cells are interesting in their own right as an efficient and effective energy conversion technology.

In this chapter, we will further introduce the context and motivation for the research (Section 1.1) and present the research questions (Section 1.2). Then we will describe the fuel cell application (Section 1.3.1), which has its own context, motivation, and research questions. Finally, we will provide an overview of the modeling approach (Section 1.4) and an outline of the dissertation (Section 1.5).

1.1 Context and Motivation

Mathematical modeling of physical systems is becoming more important due to the increasing complexity of engineered systems, the emphasis on system design, and improvements in modeling languages, tools, and algorithms. Models are used in hardware and control design to run tests more quickly and cheaply than physical experiment. They are used in research to explore hypotheses of physical behavior and to provide virtual sensors where physical sensors have lower fidelity, more uncertainty, or are simply not available or practical. Models can also be simulated in real time for model-based control and model-in-the-loop testing.

There are many types of mathematical models of physical systems and many methods of classification:

- By physical domain: electrical, magnetic, mechanical (rotational or translational), thermal, fluid, chemical, etc.
- By mathematical formalism: algebraic equations, ordinary differential equations (ODEs), differential algebraic equations (DAEs) or partial differential equations (PDEs)
- By mathematical complexity: linear or nonlinear
- By mathematical causality: causal or acausal
- By the inclusion of time: static or dynamic
- By spatial dimensionality: zero-dimensional (0D), one-dimensional (1D), or multi-dimensional
- By the representation of time: discrete, continuous, or hybrid
- By the representation of space: discrete, continuous, or hybrid
- By the level of physical abstraction: physics-based or empirical
- By encapsulation: flat or modular
- By the representation of physical hierarchy: flat or hierarchical
- By the programming or modeling language: C, Java, MATLAB, Modelica, etc.

The choice of the type of model to use for an application depends on many factors including features of the physical system, what needs to be determined by the model and with what accuracy, how much the model will be reused, the cost of modeling and simulation software, the available computational resources, and the existing models and literature.

Due to the increasing complexity of engineered systems and the emphasis on system design, it is helpful if a model is modular, hierarchical, and acausal like a real system. Modularity allows a modeler or designer to more easily reconfigure a model to consider multiple physical architectures. Hierarchy allows detail to be hidden or encapsulated, without loss, as a system becomes more complex. With modular, hierarchical, and acausal features, a model can convey not only the equations that govern behavior but also the structure of the system in a way that is useful for both human understanding and computer interpretation for simulation and other processing.

Acausal models and modeling languages are also called *declarative*, equation-based, or physical-interaction [6]. Declarative language preserves the simultaneous mathematical nature of equations. By definition, an equation declares a relation between two expressions without implying mathematical causality (i.e., assigning independent and dependent variables). The *causal* approach is also called *imperative*, sequential modular [7], or signal-flow [6]. It relies on assignment operations which are organized to form algorithms with predetermined input/output assignments. The advantages of declarative language are described in detail below.

1.1.1 Advantages of Declarative Modeling

Declarative language has four main advantages over causal or imperative language in modeling physical systems. Declarative models are true to the acausal nature of physics, and compared to imperative models, they are more intuitive, more flexible and reusable, and less prone to user error. These advantages are presented in more detail in the following paragraphs.

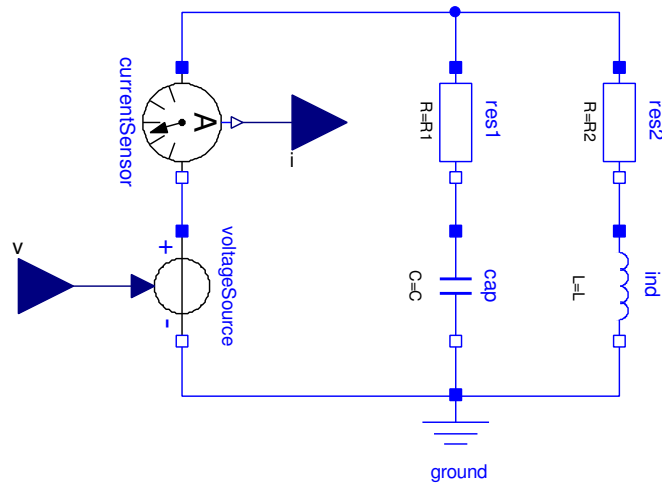
The first advantage is that declarative language best represents the nature of physical behavior and preserves the meaning of physical laws [8–11]. For example, although current leads voltage in an electrical capacitor, the current does not cause the voltage to change any more than the change in voltage causes a current. Placing the causality assignment in the capacitor's physical description is to suffer from the “post hoc ergo propter hoc (it happened before, hence it caused) fallacy” [9]. Declarative language allows the relationship to be expressed directly, without causality. Imperative models should be reserved for flows of information in control engineering, signal processing, and similar fields of study. As stated by Matei and Bock, “Physical

conservation laws do not apply in these applications because the same information can flow (be copied) to multiple components, while physical things cannot.” [6]

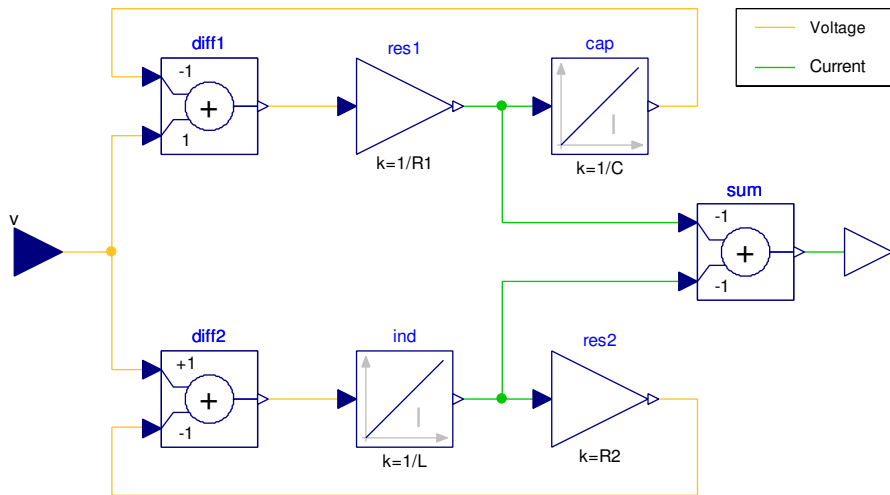
The second advantage is that declarative models are more intuitive than imperative ones. This is demonstrated by Figure 1.1, which shows declarative and imperative models of the same electrical circuit. The connections of the declarative model (Figure 1.1a) represent wires which imply Kirchhoff’s voltage and current laws (KVL and KCL). The diagram shows how the components would be actually assembled. In contrast, the connections of the imperative model (Figure 1.1b) represent signals. The type of signal (voltage or current) depends on the connection’s context within the circuit, since the block for each electrical component receives voltage and transmits current or vice versa. The topological equations (KVL and KCL) are represented by difference and summation blocks, and this distracts attention from the blocks which represent the constitutive equations of the capacitor, inductor, and resistors.

The third advantage is that declarative models are more flexible and reusable because they preserve the information necessary to perform symbolic manipulation. Powerful modeling tools exist (see Section 2.1) that can solve a model for the imposed causality, linearize a model, partition a dynamic model into the most numerically efficient systems of algebraic equations (i.e., resolve algebraic loops through tearing), and perform index reduction (i.e., eliminate structural singularities) [12, 13]. In addition, methods are being developed for analytical model order reduction (MOR) [14]. Returning to the electrical example, the declarative model of Figure 1.1a independently maintains information about the circuit (capacitor, inductor, resistors, and their connections) and the causality imposed on it by the boundary condition (voltage source). If the boundary condition is changed (e.g., current source instead of voltage source), a modeling and simulation tool can automatically change the causality as needed. However, the imperative model of Figure 1.1b must be manually re-solved and reconfigured as shown in Figure 1.2. The correlation is not obvious, which hinders model development and use. It is not practical, especially for complex systems, to maintain multiple versions of a model for the sake of causality. Automatic linearization is helpful in evaluating dynamic characteristics and in control techniques such as model predictive control (MPC). Algebraic loops tend to occur in the representations of complex physical systems; therefore, it helps if declarative modeling

tools can handle them in a robust manner. Index reduction can be used as a tool to scale the amount of detail included in a model without adding simulation overhead.



(a) Declarative.



(b) Imperative.

Figure 1.1: Imperative and declarative and imperative models of an electrical circuit [15].

The fourth and final advantage of declarative models is that they are less prone to user error. Some modeling mistakes may be avoided because the conservation laws are inherently and rigorously included. If the developer of a model library uses the correct conservation equations in the lowest-level models, any circuit a user builds from the library is guaranteed to also include the correct conservation equations via Kirchhoff's current law, which is applied at every node. In an imperative model, the conservation equations must be manually included

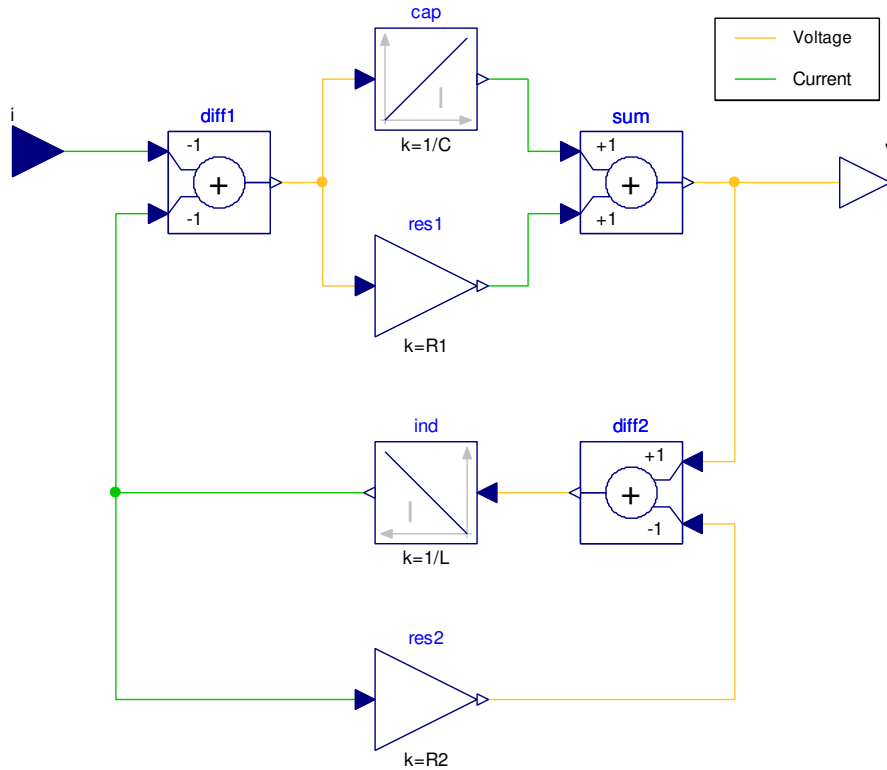


Figure 1.2: Inverse imperative model of the circuit in Figure 1.1.

at every level of the model; therefore, it is easier to violate conservation equations [6]. In addition, a user may be tempted to cascade two instances of imperative models such as the ones in Figures 1.1b and 1.2. This is incorrect because the impedance of the second circuit affects the output of the first. Two instances of the declarative model (Figure 1.1a) can be easily and correctly cascaded by connecting the two in parallel (after removing the voltage source of the second instance).

1.1.2 Current Limitations

Equation-based, object-oriented (EEO) or declarative, circuit-based modeling has been gradually extended from Kirchhoff's electrical circuit laws of 1845 to the magnetic, rotational, translational, thermal, fluid, and chemical domains. In each domain, the *efforts* and *flows*, or physical quantities analogous to voltage and current, are now well-established. However, the fluid and chemical domains are more complicated because the material flow carries not only

atoms or molecules but also significant amounts of other conserved quantities such as momentum and energy [16]. This process is called *advection* and is depicted in Figure 1.3a. By nature, the amount of these quantities depends on the source of the material. Methods have been developed to use the property of the source by switching according to the direction of material flow (see Section 2.2).

However, the switching approach has two drawbacks. The first is that it generally requires reinitialization upon flow reversal. This can usually be handled without a problem, but it requires additional computation. In some cases there is chattering, or frequently repeated flow reversal, which can slow a simulation considerably. The second drawback is that the intensive property is ill-defined when the material flow rate is zero. This can be addressed by building a method of regularization into the switching algorithm. This is of little immediate consequence because there is no advection at zero material flow rate. However, it is at zero material flow rate that *diffusion*, or transfer due to thermal activity with no net material flow, dominates. Diffusion occurs towards lower values of the intensive property as depicted in Figure 1.3b. Diffusion is not captured by the switching approach.

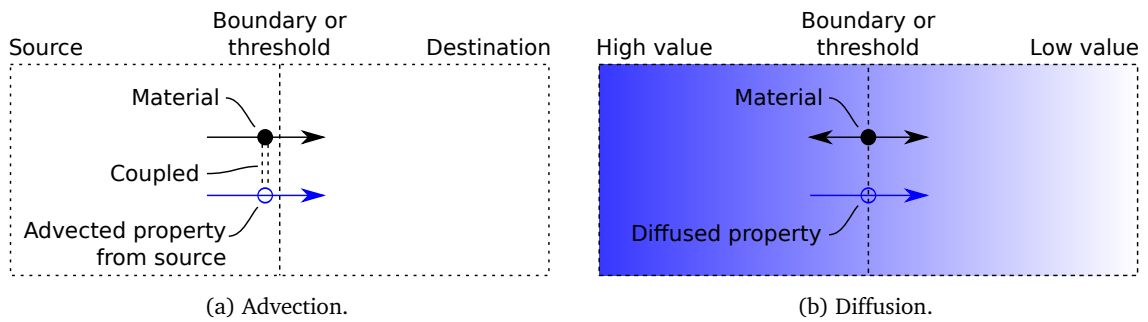


Figure 1.3: Depiction of the two fundamental modes of transfer.

In some physical situations, there is mixed advection and diffusion. To describe these situations, it is possible to add an additional pathway for purely diffusive transfer. However, this is redundant and inconsistent because it yields two intensive properties at the boundary—one for advection and one for diffusion. It also does nothing to eliminate the switching behavior or to resolve the advected property which is ill-defined at zero material flow rate. This is the problem that the present research addresses—to model coupled advection and diffusion in a

manner that is declarative, object-oriented, mathematically well-defined, numerically efficient, and physically representative yet generic.

1.2 Research Questions

The goal of this research is to realize the advantages of declarative modeling for complex physical systems that involve both advection and diffusion to varying degrees in multiple domains. We seek to answer the following questions:

RQ1: How can we create a generic declarative framework to model systems with processes that exhibit coupled advection and diffusion?

RQ2: How can the equations be best implemented to reflect the physical structure of a device and support reconfiguration?

RQ3: How appropriate is the framework for modeling all the relevant physical phenomena of an electrochemical device such as a fuel cell?

The last question will be elaborated in the following section.

1.3 Application to Fuel Cells

In this dissertation, the modeling framework will be applied to fuel cells. This will provide a pertinent and nontrivial demonstration of the framework while also establishing a novel approach to fuel cell modeling. Declarative fuel cell models are physically appropriate [8], yet few models of this type exist (as shown in Chapter 2).

1.3.1 Context and Motivation

Fuel cells (FCs) have the potential to serve a key role in our electric power networks, transportation systems, and portable electronic devices. In general FCs can convert fuel energy to work more efficiently and quietly than internal combustion engines (ICEs) [17], and a FC system's energy-to-power ratio can be easily adapted, unlike batteries. A FC system can be refueled quickly like an ICE system, or it can be designed to recharge like a battery by operating in electrolysis mode [18]. Of the various fuel cell technologies, proton exchange membrane fuel cells

(PEMFCs) are best suited to meet the packaging and power-cycling requirements of vehicles and portable devices.

PEMFCs have a solid polymer-based electrolyte and operate at low temperatures (typically below 100 °C). As shown in Figure 1.4, a single-cell PEMFC has few main components: a proton exchange membrane (PEM), two catalyst layers or electrodes, two gas diffusion layers (GDLs), and two flow plates (FPs) [17]. However, most applications require a higher voltage than a single-cell PEMFC can provide; therefore, two or more cells are joined back-to-back to form a PEMFC stack like the one shown in Figure 1.5.

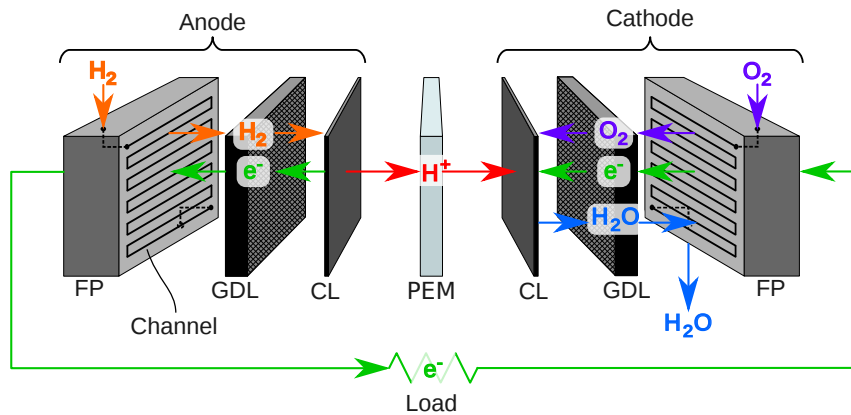


Figure 1.4: Layers of a single-cell PEMFC and the primary paths of electrons (e^-), protons (H^+), hydrogen (H_2), oxygen (O_2), and water (H_2O) during normal operation.

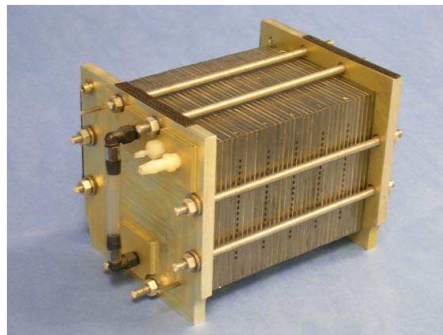
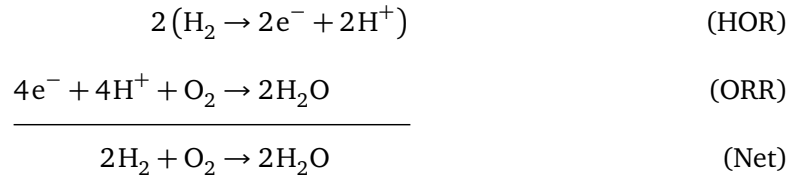


Figure 1.5: 500 W, 32-cell, water-cooled PEMFC used to power an unmanned aerial vehicle [19].

A PEMFC operates on the chemical energy released by the reaction of hydrogen (H₂) and oxygen (O₂) to produce water (H₂O):



Its PEM (electrolyte) controls the reaction by selectively passing protons while acting as a barrier layer to hydrogen, oxygen, and electrons (e⁻), as shown in Figure 1.4. This forces the reaction to occur in two sub-reactions: the hydrogen oxidation reaction (HOR) whereby hydrogen is consumed and electrons and protons (H⁺) are produced, and the oxygen reduction reaction (ORR) whereby oxygen, electrons, and protons are consumed and water is produced. In order to complete the full reaction, the electrons must travel an external path. The path is provided by an external load which can harness the energy of the net reaction.

However, the cell has internal losses that cause heat generation and limit the rate at which the electrons can do a given amount of external work. Some of the energy goes into making the reactions occur fast enough (activation losses), friction between the charge carriers (e⁻ and H⁺) and the conducting materials (Ohmic losses), and transporting the reactants to and products away from the catalyst layers (concentration losses).

In order to operate, a PEMFC must be supported by other components, which are collectively called the balance of plant (BOP). The PEMFC stack and BOP are the basis of a complete PEMFC system like the one shown in Figure 1.6. Figure 1.7 shows the configuration of two PEMFC systems—one relatively simple and the other relatively complex. Even more complex PEMFC systems may include heat recovery [20] and fuel reformation.

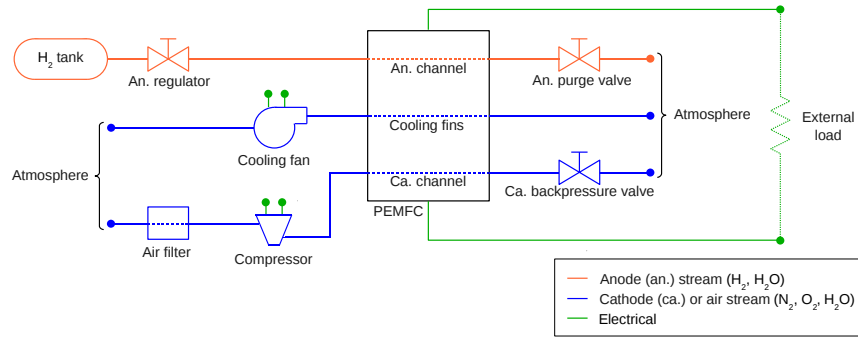
Although PEMFC systems are promising, their cost and durability, and to a lesser extent, size and weight, must be improved to meet the desired standards for commercialization [21, p. 11]. The U.S. Department of Energy has outlined the numerous avenues that are being pursued to improve the PEMFC, BOP components, and PEMFC system as a whole [21]. The physical mechanisms of PEMFC degradation and failure are being determined and characterized through experimental and model-based investigations [21, pp. 3, 9, 32, & 40]. Novel materials



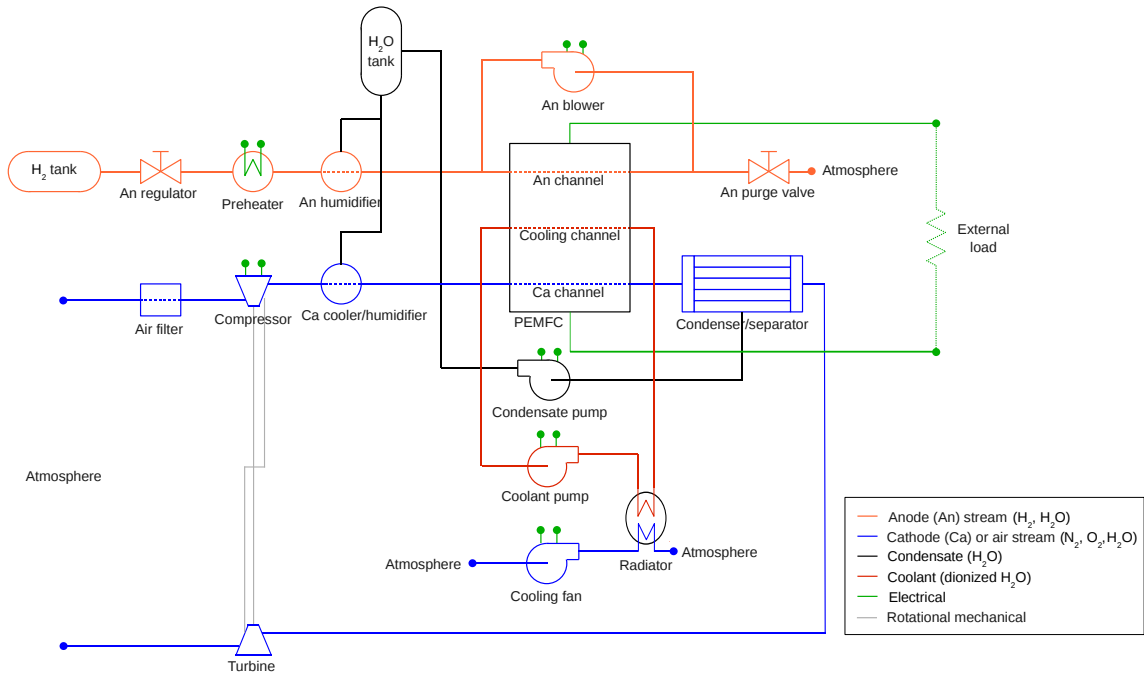
Figure 1.6: 100 kW PEMFC system used to power a bus. From Georgetown University, “Generation III Project,” <http://fuelcellbus.georgetown.edu>, accessed Nov. 2011.

and structures are being identified and developed for the core components of a PEMFC (Figure 1.4) as well as the seals between and around them [21, pp. 4–7]. These efforts seek to lower the cost of materials, improve thermodynamic efficiency (by decreasing the activation, Ohmic, and concentration losses), and improve robustness (specifically to air and fuel impurities, temperature and humidity variations, corrosive conditions, and power cycling). Design techniques and manufacturing processes are being developed to support the low-cost and high-throughput production of PEMs, electrodes, and flow plates [21, pp. 2, 5–6, & 29–30], [22]. One goal is to more effectively integrate the PEM and electrodes (as a membrane electrode assembly) in order to minimize interfacial resistances, while at the same time allowing the raw materials to be reused or recycled [21, pp. 3–10].

The BOP components are being improved as well. New materials and concepts are being applied to heat exchangers, humidifiers, compressors, and turbines with the goal of reducing their size, weight, and cost [21, pp. 7, 10, & 32]. Since the air compressor places a significant internal electrical load on the PEMFC system, it is important to maximize its efficiency, and in some cases, a turbine is beneficial [17, p. 102]. Air filtration technology is being evaluated to allow PEMFCs to be used in off-road applications. Sensors, especially those for chemical composition, are being further developed for reduced cost and size and improved accuracy, reliability, durability, and dynamic response. [21, pp. 7]. The design of reformers and the



(a) Simple.



(b) Complex.

Figure 1.7: Configurations of two hydrogen-fueled (non-reforming), pressurized PEMFC systems where (a) the stack is air-cooled and (b) the stack is water-cooled, the anode is preheated and recirculated, the anode and cathode are both humidified, and the cathode exhaust is expanded through a turbine.

operation of PEMFC on reformat is also being improved [21, pp. 8–9], but in applications fueled by hydrocarbons rather than hydrogen, PEMFCs are less likely to be prevalent over high-temperature FC technologies that can accept those fuels directly (and perform internal reformation).

The final area of work considers how to improve the PEMFC system as a whole. Alternative PEMFC system configurations and design parameters are being considered that may allow the functions of the PEMFC system to be performed by fewer or simpler components, or that may entirely eliminate the need for certain functions [21, p. 10]. For example, one of eight or more possible methods for external humidification may be chosen, or the PEMFC can be operated within certain ranges of temperature and air flow rate where external humidification is not necessary [17, pp. 83–90]. Such choices must be guided by sufficiently accurate information, so PEMFCs are being tested to evaluate their performance, durability, and other properties under various operating conditions, including hydrogen impurity [21, p. 9].

Mathematical models are used to assist many of these efforts to develop PEMFCs. These models offer several benefits. First, the operating conditions of a model can (in theory) be adjusted and measured easily. As stated by Cellier, “in the simulation world, all inputs and outputs are accessible” [23]. This can help provide insight into working mechanisms of a fuel cell [20] via techniques such as model-based data analysis [24]. Second, simulations are perfectly repeatable. Models are not subject to unidentified disturbances and measurement error. This means, for instance, that the effects of a design parameter can be clearly identified. Third, fuel cell models are faster and cheaper to run than test equipment [20, 24]. This is very important in design exploration, where many tests must be performed. It also allows extreme operating conditions to be tested without risking damage to the fuel cell hardware. Fourth, fuel cell models can help to organize and share knowledge about the configuration of a fuel cell and its working principles [24]. This is particularly important since fuel cells are multi-physical devices that require multi-disciplinary research and development.

However, due to the complexity of the structures and the physical processes that occur within PEMFCs, specialized models are typically required for different situations. Many academic articles have been published with PEMFC models that are appropriate and useful for

particular cell designs, operating conditions, and levels of fidelity (i.e., spatial, dynamic, or behavioral detail [25]) [20].ⁱ Ideally, variants of a common PEMFC model could be used for a wide range of research and development work, including physical investigation, model-based systems design, and model-based control. Such a model library could offer an open framework for PEMFC researchers to contribute their expertise and benefit from the collective knowledge of others.

A broadly applicable PEMFC model library would need to contain models that are *physically representative*, meaning their predictions of behavior match reality (i.e., *accurate*) and their structure corresponds to the physical domain. Specifically and at a minimum, the static voltage-current predictions should be accurate over the following ranges of operating conditions:

- 0.3 V to 0.9 V electrical potential difference (per cell)ⁱⁱ
- 20 °C to 80 °C temperatures of flow plates and inlet gases (all varied together)ⁱⁱⁱ
- 1 atm to 3.5 atm absolute anode/cathode outlet pressures (varied together)^{iv}
- 0% to 100% relative humidity at anode inlet^v
- 30% to 70% relative humidity at cathode inlet^{vi}
- 14% to 100% mean inlet/outlet oxygen concentration in dry cathode gas^{vii}

The PEMFC model library should approximate the dynamic voltage-current response of actual cells at nominal operating conditions and varying large-signal electrical currents (e.g., [28]). It should capture the operational effects of design parameters including component sizes and material properties (for hardware analysis and design) and should be capable of linearization

ⁱThe next chapter contains a literature review.

ⁱⁱLow cell potentials (high electrical currents) are avoided in order to reduce the chemical/electrochemical transport losses. High cell potentials are avoided because they accelerate the corrosion that occurs due to electrical cycling [26, pp. 6–7].

ⁱⁱⁱThe lower bound corresponds to start-up from room temperature. Nafion, the most common membrane material, dehydrates above ~ 80 °C, which increases its protonic resistance [27].

^{iv}Some PEMFCs operate at atmospheric conditions. PEMFC system efficiency is unlikely to increase above a pressure ratio of ~ 3.1 [17, p. 107]

^vFor simplicity, some PEMFC systems do not have humidifiers (0%). Other systems humidify the anode as much as possible without causing flooding to occur (up to 100%).

^{vi}The lower bound (30%) corresponds to the relatively extreme case of operating the PEMFC without humidification in the Sahara desert on an average day [17, p. 78]. The cathode supply is usually not saturated; flooding would occur since H_2O is also produced in the cathode.

^{vii}The lower bound corresponds to a stoichiometric ratio of 1.5. The risk of starvation increases as the ratio approaches 1.0. PEMFCs are also operated on pure oxygen in applications such as unmanned underwater vehicles (UUVs).

(for control analysis and design). It should be able to describe relevant phenomena including electrochemical reactions, chemical/electrochemical transport, heat transport, and heat generation. It should also have variable fidelity so that it can be used for layer-, cell-, stack-, system-, or application-level simulations. Finally, it should be modular, meaning it should be possible to interconnect its submodels in various ways to build larger models analogous to the physical hierarchy. Unfortunately, no current PEMFC model library can provide these features, let alone over the required range of operating conditions.

1.3.2 Research Questions

As stated in research question three (RQ3), we will investigate how appropriate the declarative advective/diffusive framework is for modeling all the relevant physical phenomena of a fuel cell. The hypothesis is that the framework can be used to help establish a fuel cell model library that is physics-based, modular, reconfigurable, accurate, and leads to numerically efficient and robust models. As discussed in the previous section, such a library would be a valuable tool for fuel cell research and development. In order to answer RQ3, we will address the following subquestions:

RQ3a: For which processes is it necessary to model mixed advection and diffusion? Where is it appropriate to assume pure advection or pure diffusion?

RQ3b: What characteristics do the models exhibit that would not be present given an imperative formalism?

RQ3c: Which combinations of accuracy and speed can be achieved by adjusting fidelity?

1.4 Overview of the Modeling Approach

As mentioned previously, the modeling approach is declarative, modular, and hierarchical. This approach is also called EOO modeling. The Figure 1.8 illustrates that the models are created by building species (e.g., H_2) into phases such as gas, phases into subregions, subregions into regions such as a fuel cell layer, and regions into assemblies such as a fuel cell. This reflects the physical architecture of the device.

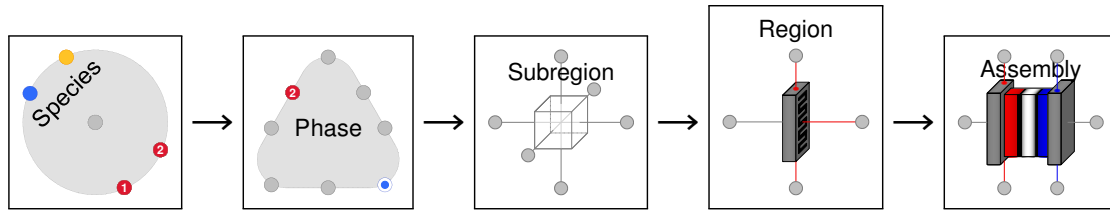


Figure 1.8: Levels of physical hierarchy in the model library.

The models are highly reconfigurable. Assumptions may be applied that affect the spatial resolution and the included species, phenomena, axes of material translation, and boundaries. With each assumption, the number and complexity of the equations scale appropriately and without simulation overhead. The characteristics of individual species are provided in replaceable packages. The packages can be used to model incompressible and compressible fluids including ideal and real gases. The thermodynamic properties and other correlations are adjusted automatically according to these assumptions.

The models are dynamic and continuous in time. Transients are modeled in terms of DAEs, or implicit ODEs combined with algebraic constraints. These DAEs are implemented in the Modelica language [1].

The models are discrete in space. As stated by Mattiussi [29], this representation has three advantages: (1) it provides a unified perspective that is appropriate for many theories, (2) it directly correlates the discretization of the physical region and the structural properties of the applied theories, and (3) it is based on intuitive geometrical and physical concepts that help distinguish the numerical methods (e.g., finite difference method (FDM), finite volume method (FVM), and finite element method (FEM)) and the underlying theories. Rather than implementing approximations to traditional PDE representations, the approach is to distill the key concepts from equations such as Navier-Stokes and formulate them in a manner best uses features of EOO language. The models have options for one, two, or three dimensions. The grid of control volumes is rectilinear but the lengths are adjustable.

Computational efficiency is emphasized. The translated models contain only linear systems of equations. Techniques are used to reduce computational complexity, for example by representing high-order polynomials in nested form. Many optional assumptions may be enabled to

simplify the model. For example, the temperature of different phases may be constrained to be the same; this results in index reduction and a simpler model.

The advective/diffusive framework is applied to the electrical, thermal, fluid, and chemical domains. The approach is deeply physics-based. It employs dynamic conservation equations for material, translational momentum, and energy. Since the models are also resolved down to the species level, this requires traditional equations to be described at a more fundamental level. For instance, Ohm's law and the Maxwell-Stefan equations for multi-component diffusion are not implemented directly. Instead, drag interactions are modeled in a manner that results in those equations.

1.5 Outline of the Dissertation

In this chapter, we have introduced the research by discussing the motivation, questions, and general approach. The subjects of the remaining chapters are as follows:

- **Chapter 2 — Background:** Survey of the relevant literature in the areas of declarative modeling languages, approaches to fluid flow in declarative language, and fuel cell models
- **Chapter 3 — Fundamentals of the Model:** Detailed presentation and justification of the model equations which cover thermodynamics, material properties, transport, and exchange
- **Chapter 4 — Implementation of the Model:** Summary of the implementation of the equations in a fuel cell model library
- **Chapter 5 — Basic Examples:** Discussion of the conditions and results of several low-level demonstrations of the model library
- **Chapter 6 — Simulation of the Fuel Cell Model:** Discussion of the conditions and results of polarization tests and a dynamic example of the fuel cell model
- **Chapter 7 — Conclusions:** Recapitulation of the dissertation, list of contributions, and suggestions for future work
- **Appendix A — Related Theory:** Derivations and discussions that relate the model to selected theories in fluid dynamics and solid-state physics

- **Appendix B — Selected Documentation and Source Code:** User documentation, diagrams, source code, and tables of parameters and connectors of the fuel cell model library

BACKGROUND

In this chapter, we will review the literature and current developments in several related areas. First, we will consider the equation-based, object-oriented (EOO) languages that are candidates for demonstrating the modeling contributions of this dissertation (Section 2.1). Then we will describe the recent work to model fluid and chemical systems using the Modelica language in particular (Section 2.2). Finally, we will review models of proton exchange membrane fuel cells (PEMFCs) with an emphasis on the modeling formalism (Section 2.3).

2.1 Equation-Based, Object-Oriented (EOO) Modeling Languages

There are five major domain-neutral, EOO modeling languages in current use and development: Modelica, VHDL-AMS, Verilog-AMS, gPROMS, and Simscape. A brief overview of these languages is given in the following sections. All the languages support differential algebraic equations (DAEs) and conservation equations via the generalized Kirchhoff current law. The differentiation is in their syntax, semantics, additional features, and available model libraries and tools. Since syntax and semantics are somewhat subject to preference, we will focus only on the features and the available libraries and tools.

Modelica, VHDL-AMS, and Verilog-AMS are standardized and tool-neutral, meaning that they are supported by multiple vendors whose software should operate on the same models. This can help to prevent tool lock-in, or dependence on a particular vendor. It also tends to encourage a community of open-source model developers. The languages of gPROMS and Simscape are proprietary and integral to the modeling platforms of the same name by Process Systems Enterprise Limited and The MathWorks Inc., respectively.

In addition to the major domain-neutral EOO modeling languages, there are several languages that offer capabilities or approaches that are not yet mainstream. Sol is a domain-neutral EOO language that is based on Modelica and supports variable-structure systems [30].

This is important, for example, in mechanics where connections may be made or broken during the course of a simulation. Hydra is the most mature language that uses functional hybrid modeling (FHM) [31]. This approach is based on functional programming languages such as Haskell. Functional programming is a method of declarative programming, so it is naturally appealing for declarative or equation-based modeling.

Many other languages and platforms exist that are not simultaneously equation-based, object-oriented, and domain-neutral. Languages such as Hybrid χ and ASCEND are declarative but do not yet support declarative or acausal connections [31].ⁱ Engineering Equation Solver (EES; by F-Chart Software, LLC) is a declarative modeling tool that has an extensive thermodynamics library, but it does not support declarative connections either. Numerous languages and platforms are declarative but are domain-specific—for example, aspenOne (Aspen Technology Inc.) for chemical process simulations.

The background of Modelica and Simscape is more general and multi-physical than VHDL-AMS, Verilog-AMS, and gPROMS. Although all five languages are neutral with respect to physical domains, the history is important because it has implications on the extent of the user base and the depth and breadth of the available tools and model libraries. Table 2.1 lists the absolute and relative numbers of articles that reference the languages in four major scientific and engineering databases: Compendex and InSpec (<http://www.engineeringvillage.com/>), ScienceDirect (<http://www.sciencedirect.com/>), and Web of Science (<http://apps.webofknowledge.com/>). In addition, relative internet search interest is listed from Google Trends (<http://www.google.com/trends/>). Modelica appears to dominate these indices; the smallest margin (12%) is with gPROMS in ScienceDirect.

2.1.1 Modelica

Modelica [1] was designed as a “multi-formalism, multi-domain, general-purpose modeling language” [32]. The designers sought to unify the basic syntax and semantics of many modeling languages that were present in the late 1990s, including Dymola, Omola, NMF,

ⁱThis aspect of ASCEND is discussed at http://www.ascend4.org/ascend_processmodeling.htm (accessed Aug. 24, 2013).

Table 2.1: Relative occurrence of declarative modeling languages.ⁱⁱ

Name	Compendex 2013		InSpec 2013		ScienceDirect 2013		Web of Science 2013		Google Trends 2008–2013	
	#	Rel.	#	Rel.	#	Rel.	#	Rel.	—	Rel.
	Modelica	108	66%	36	55%	112	48%	42	64%	74
VHDL-AMS	22	13%	10	15%	15	6%	9	14%	11	9%
Verilog-AMS	4	2%	6	9%	9	4%	2	3%	3	3%
gPROMS	11	7%	3	5%	84	36%	7	11%	13	11%
Simscape	18	11%	10	15%	13	6%	6	9%	17	14%

SIDOPS+, Smile, ObjectMath, ASCEND, and U.L.M. [32, 33]. Of these, gPROMS, VHDL-AMS, and ASCEND are still independently active. Dymola is now a modeling and simulation tool that supports Modelica.

The Modelica language specification is still evolving with new releases every year or two. It includes keywords and operators for discrete as well as continuous systems. Methods have been proposed to support partial differential equations (PDEs) [34, 35], but they have not yet been integrated into the language [1]. Meanwhile, an imperative block-based library is available that uses the method of lines (MOL) or the finite volume method (FVM) to convert parabolic or hyperbolic PDEs to ordinary differential equations (ODEs) [36, 37].ⁱⁱⁱ

Modelica is supported by a growing number of software tools including Dymola (Dassault Systèmes), SystemModeler (Wolfram Research), MapleSim (MapleSoft), AMESim (Siemens AG), CyModelica (CyDesign Labs), SimulationX (ITI GmbH), and MWorks (Suzhou Tongyuan). In addition, there are free modeling and simulation environments including OpenModelica (Open Source Modelica Consortium) and JModelica (Modelon AB).

Modelica has more than thirty free, open-source libraries contributed by the user community.^{iv} The Modelica Association maintains the Modelica Standard Library, which contains well-established packages of thermal, fluid, electrical, magnetic, and mechanical components. It also contains a package of thermodynamic and transport properties.

ⁱⁱThe percentages have been rounded and may not add to 100%.

ⁱⁱⁱAvailable as the PDELib package at <https://github.com/modelica-3rdparty/PDELib>.

^{iv}See <https://modelica.org/libraries>.

2.1.2 VHDL-AMS

VHDL-AMS is the combination of VHDL, a modern hardware description language, and extensions for analog and mixed signals.^v Whereas hardware description languages (HDLs) have been used to describe the behavior of physical devices and processes since the 1960s, modern HDLs also describe the structure of the device. VHDL itself is an equation-based language for digital circuits in discrete time with events [38].

The analog and mixed signal extensions are not specific to the electrical domain [38, 39]. However, due to the history of VHDL, the extensions are particularly appealing in cases where it is desirable to incorporate a model of the physical system with a digital circuit (e.g., an application-specific integrated circuit (ASIC)).

2.1.3 Verilog-AMS

Verilog-AMS parallels VHDL-AMS in many aspects. VHDL and Verilog are both HDLs, and both have been extended for analog and mixed signals. Verilog-AMS was developed and is maintained by the Accellera consortium [40] but has not yet been standardized by IEEE like VHDL-AMS. Verilog-AMS does not support replaceable or encapsulated models like Modelica and VHDL-AMS [41]. The equations in an analog block of Verilog-AMS must be manually sorted, and implicit equations are not entirely supported [41]. Pêcheux et al. [41] compare Verilog-AMS and VHDL-AMS in detail.

2.1.4 gPROMS

The gPROMS^{vi} language was created in 1992 to support combined discrete/continuous systems [42]. Several years later it was extended for partial differential equations using the MOL [43]. gPROMS is now a commercial product of Process Systems Enterprise (PSE).

The gPROMS software suite is primarily marketed and applied to chemical process modeling [31]. It has built-in tools for parameter estimation. There are various add-on modules and

^vVHDL-AMS stands for very-high-speed integrated circuit (VHSIC) hardware description language with analog and mixed-signal extensions.

^{vi}gPROMS stands for general process modeling system.

libraries for chemical/physical properties and advanced components. A fuel cell model library is also available.^{vii}

2.1.5 Simscape

Simscape [44, 45] extends the Simulink control systems platform with support for physical system simulation. The Simscape language is a relatively new offering from The Mathworks Inc. (October 2008) [46] and is likely the company's response to the predominance of Modelica. Its syntax is somewhat similar to Modelica but is not based on an open standard. Simscape includes libraries of mechanical, electrical, thermal, and hydraulic components which are similar in concept to those of the Modelica Standard Library. However, they are integrated with the product rather than freely available.

2.2 Fluid/Chemical Modeling in Modelica

Much of the literature in fluid dynamics and mass transfer uses PDEs, but the core formalism of EOO models is differential algebraic. PDEs may be introduced by extensions to the EOO language (e.g., gPROMS [43]) or model libraries (e.g., Modelica [1]); however, the capabilities are limited compared to languages and tools that are designed primarily for PDEs (e.g., OpenFOAM or COMSOL). Thus, it may be best to implement analytical solutions or correlations to experiments or offline computational fluid dynamics (CFD) simulations where possible. Co-simulation is also becoming a strong option with the movement towards software interoperability (e.g., the functional mock-up interface—FMI). However, in using co-simulation (as well as PDE extensions or libraries in an EOO environment), it is important to weigh the value of increased fidelity against the computational requirements of distributed submodels in an otherwise lumped-parameter model.

Given the current limitations of including PDEs in an EOO environment and the conceptual advantages of discrete-space representations (see Section 1.4), the following survey is limited to DAE-compatible representations of fluid and chemical systems. It is also limited to the Modelica language, since it offers the largest base of open-source work in the area. The discussion

^{vii}See <http://www.psenterprise.com/gproms/aml/fc> (accessed Aug. 24, 2013).

emphasizes advection and diffusion since they are central to fluid and chemical models. A common theme is how the one-way nature of advection is handled given the two-way nature of equations and declarative language.

2.2.1 Without Coupled Advection

The easiest approach to modeling fluid/chemical systems is to ignore the advection of properties with the material stream. In some physical situations, this is appropriate because the advection of momentum and energy is insignificant and the material flow is purely advective or diffusive. For example, in chemical diffusion, the material flow may be slow enough that the rates of advection of momentum and energy are negligible or are dominated by translational and thermal diffusion (i.e., friction and thermal conduction).^{viii} These assumptions are implicit in the BioChem [47, 48] and ADGenKinetics [49] biochemical libraries. Their connectors only consist of concentration as an effort variable and molar flow rate or volumetric reaction rate (respectively) as a flow. In isothermal, isochoric, pressure-driven scenarios (e.g., simple pipe flow), there is no material diffusion. Heat transport may not be of interest, and the rate of momentum transport (dynamic pressure times area) is proportional to the square of the rate of material advection. Then it is only necessary to include the pressure and material (or mass) flow rate at each boundary. This is the case in pure hydraulics (e.g., the OpenHydraulics library in Modelica [50]). It is also essentially the case for electrical circuits, where electrical conduction (a la Ohm's law) is actually material advection (of charge carriers).

2.2.2 Central Difference

Another approach is to include advection but ignore its one-way nature. This is the essence of the central difference scheme. It can be accomplished in an EOO framework by adding a diffusive pathway for the appropriate quantities (e.g., energy) to determine properties such as temperature at a boundary. If the resistances to either side of a boundary are equal, then the value of a property at the boundary is the mean of the values in the neighboring subregions,

^{viii}As presented in Chapter 3, purely diffusive material flow can cause thermal advection. It should be noted that thermal convection is thermal conduction in series with thermal advection.

just as it is in the central difference scheme. Advection can then be determined using the mean value.

Unfortunately, the central difference scheme gives unrealistic results when the rate of advection is large compared to the rate of diffusion [51]. It is generally accepted that some form of upstream discretization is necessary, and current Modelica libraries do not use the central difference scheme for fluid flow.

2.2.3 Without Coupled Diffusion

The third approach is to assume that the advection is not accompanied by any diffusion. If there is no diffusion, then the advected property only depends on the material source(s). This is the upwind scheme, also known as the upwind-difference scheme, the upstream-difference scheme, or the donor-cell method [51]. Unfortunately, the upwind scheme implies causality which is difficult to implement in declarative or acausal language. It introduces challenges with respect to (1) the semantics of the language and (2) the numerical robustness and computational efficiency under initialization, zero material flow, and flow reversal. These challenges have led to a number of sub-approaches, which are described in the following sections.

Due to the assumption that the advected properties are not affected by diffusion, this approach is more appropriate for fluid networks than for chemical devices. Diffusion could be added in parallel with the advective flow, but as mentioned in Section 1.1.2, this would produce redundant and inconsistent intensive properties at a boundary.

2.2.3.1 *Balanced Efforts and Flows*

One sub-approach, embodied by the `semiLinear` function of Modelica, is to implement the upwind scheme using pairs of effort and flow variables [52]. This is appealing because the effort/flow construct is well-established in EOO languages and supports an arbitrary number of connections at a node. The advected property is like an effort in that it is equal among the connected models. It can be calculated from the generalized Kirchhoff current law for the associated flow variable.

The value of the advected property depends on the material source(s), so it is discontinuous upon reversal of the material or mass flow. This is not generally a problem, but difficulties arise

when the material or mass flow rates are zero or are not explicitly known. If the mass flow rates are zero, then the effort variable (the advected property) disappears from the system of connection equations—a mathematical singularity. If the mass flow rates are not known, they must be solved from the connection equations. This is challenging because it is a nonlinear problem with Boolean expressions (the upwind conditions) [53].

2.2.3.2 *Special Connectors*

Other implementations add special variables to the connectors besides the primary effort/flow pair for material or mass transport. The conservation equation associated with the advected property is instantiated multiple times for each node. This is in contrast with the balanced effort/flow approach (previous section), where there is one conservation equation for every node—the generalized Kirchhoff current law.

The standard approach in the Modelica language since version 3.1 (May 2009) is to use a connector where the `stream` keyword denotes a property which is advected with the material. This property is not an effort or a flow, and in fact, it is not equal among connectors at a node. A stream connector contains one driving property such as pressure, one flow variable such as mass flow rate, and one or more stream properties. Model equations can reference a stream property directly or via built-in `inStream` or `actualStream` operators. A direct reference yields the value of the property that would hypothetically occur if material is exiting a component. The `inStream` operator returns the value that would occur if material is entering the component. The `actualStream` operator returns the actual value which depends on the flow direction [1].

This organization avoids the need to devote a variable to the actual, mixed value of the advected property. As mentioned previously (Section 2.2.3.1), that value is ill-defined at zero-flow conditions and discontinuous upon flow reversal. With the `actualStream` operator, it is determined only where it is needed, for example in the conservation equations of control volumes and in the definitions of variables for analysis and plotting. In the conservation equations, it is multiplied by the material or mass flow rate such that the product is not discontinuous [1]. In other cases, either the outflow value (direct reference to the stream variable) or the inflow

value (`inStream` operator) is appropriate. The outflow value usually depends on the thermodynamic state of a control volume (no Boolean conditions) or on inflow value(s). The inflow value is calculated from a unique conservation equation for each connector given the assumption that fluid is entering the associated component. Although the actual material or mass flow rates of the other connectors are used in the equation, this means fewer discontinuities [53]. The Modelica specification requires that modeling tools implement a method of regularization to eliminate the singularities at zero material flow [1]. However, since stream connectors are a relatively new addition to the Modelica language, some modeling tools do not fully support them.

The Modelica Fluid library, which is part of the Modelica Standard Library, uses stream connectors to model one-dimensional (1D) fluid networks [2]. Like the balanced effort/flow method (previous section), multiple stream connectors can be connected to a node.

Some one-dimensional (1D) fluid libraries use custom upwind implementations that place restrictions on the connections. Both ThermoSysPro [54] and ThermoPower [55] each contain two basic types of fluid connectors. A connection must consist of exactly one connector of each type; therefore, a stream can only be split or merged using special models [53]. Essentially, a conservation equation is implemented for each of the two possible flow directions, but only one is selected at a given time. ThermoSysPro uses only effort variables (no flows). In fact, “no physical meaning is assigned to the fluid connectors: they are considered as a means to pass information between components, so they are not part of the physical equations” [54]. Since Modelica 3.0 (Sep. 2007) [1], this approach has been outlawed in order to improve model quality [56], but it is still allowed by some tools. ThermoPower uses opposing inputs and outputs to transmit information in the two possible downstream directions. The correct signal is chosen depending on the direction of material flow [53,55]. Unfortunately, the ThermoPower pipe models do not guarantee material conservation due to the method used to discretize the underlying PDEs.

All of these libraries—Modelica Fluid, ThermoSysPro, and ThermoPower—use a staggered grid approach. The dynamics of translational momentum, if included, are inside the pipe models. The material and thermal dynamics is included in the volume models. Typically, a fluid

network consists of alternating volumes and pipes or other restrictive devices. The staggered grid approach is one way of avoiding a wavy pressure and velocity profile along a flow path, which is an unrealistic model result [51].

The Modelica Fluid, ThermoSysPro, and ThermoPower libraries use the Modelica Media library to varying degrees. The Modelica Media library represents thermodynamic and transport properties of a large variety of fluids. The concept is to use replaceable classes to describe the fluid properties within the hardware models (vessels, pipes, etc.). This serves two purposes: to enhance the flexibility of the hardware models and to lessen the barrier to creating new hardware models. There are currently two main drawbacks: (1) the overhead of supporting all the necessary ways of accessing the same information and (2) the fact that chemical species are not independently selectable.

2.2.3.3 Bond Graphs

Another sub-approach of the upwind scheme is to use coupled bond graphs. In bond graphs, there are no efforts and flows—not even for material or mass transport. The built-in mechanism to generate the Kirchhoff circuit laws (`connect`) is not used [57]. Instead, junction models are used to implement these equations.

Cellier et al. have developed libraries to model thermodynamic systems and chemical networks using these coupled bond graphs or “multi-bonds” [16,58,59]. The bonds may be causal or acausal. The connectors include multiple effort variables but no flow variables. As mentioned previously, this approach is now illegal according to the Modelica language specification.

ThermoBondLib, the thermodynamic library of Cellier et al., uses media models with relatively simple correlations instead of models from the Modelica Media library. The thermal conjugate variables are temperature and entropy flow rate. However, thermal conduction is known to be nonlinear in this formulation [60]. Pseudo-bond graphs, which use heat flow rate instead of entropy flow rate, are often preferred [61].

The compiled models of ThermoBondLib appear to have a significant amount of algebraic overhead due to the large number of basic models and connections associated with the bonds and junctions. Another drawback of the additional models is that bond graphs are less intuitive

than the typical circuit-based form of EOO models. However, they may offer additional insight to skilled bond-graph modelers [62].

2.3 Fuel Cell Models

William Grove probably established the first fuel cell model in 1842 by describing the basic working principles of a fuel cell [63, pp. 3–4]. However, most of the recent PEMFC models can be traced to those developed by Springer et al. [64,65] and Bernardi and Verbrugge [66,67] in the early 1990s. There are well over 200 of these models.^{ix}

Some of the recent or otherwise notable physics-based and phenomenological modeling papers are discussed in Sections 2.3.1 and 2.3.2 below. Here, a model is considered physics-based if it contains a form of the Navier-Stokes equations. The equation-based, object-oriented (EOO) fuel cell models are set aside for a separate, more detailed discussion in Section 2.3.3. Only models that include electrochemistry are included below. Some papers do not include electrochemistry because they focus on fluid transport (e.g., [69]); others use neural networks instead of explicit electrochemical equations (e.g., [70], [71]).

More information is available in the reviews by Weber and Newman [68], Wang [72], Haraldsson and Wipke [73], Faghri and Guo [20], and Djilali [74]. In addition, the fuel cell modeling paper by Dawes et al. [75] begins with a very good literature review.

2.3.1 Physics-Based

Physics-based or theoretical models can be used to help explain observed behavior and evaluate hardware designs with relatively high spatial resolution. However, they have not been used directly in the design and analysis of fuel cell control algorithms due to their computational complexity. In theory, a CFD description could be linearized into state space for control analysis and design, but it is difficult to retain the essential physical behavior in the process [76].

Table 2.2 summarizes the features of some notable physics-based PEMFC models. The models all use a form of the Navier-Stokes equations to determine the velocity of the fluid.

^{ix}based on a count in 2004 by Weber and Newman [68, p. 4681]

Only Berning and Djilali [77] and Nguyen et al. [78] consider compressible flow. The only dynamic model is by Wang and Wang [79].

To model material diffusion (e.g., through the GDL), either the Maxwell-Stefan equations (coupled rates) or Fick's law (independent rates) is used. Several of the models that use Fick's law have diffusion coefficients that depend on the concentrations as presented by Slattery and Bird [80].

Most of the physics-based models use the Nernst equation to determine the open circuit voltage and the Butler-Volmer equation to determine the overpotentials of each half reaction. However, several of the models use simplifications. Sousa et al. and Mazumder and Cole use a lumped Butler-Volmer equation for the anode and cathode [81,82]. Sousa et al. heavily modify the Butler-Volmer equation [83]. Dutta et al., Chippar and Ju, Wang and Wang, and Um et al. use the Tafel equation for the cathode [79,84–87]. Sivertsen and Djilali assume that the anode overpotential is constant [88], and Dutta et al. seem to assume that it is zero [84]. Chippar and Ju, Wang and Wang, and Um et al. linearize the anode Butler-Volmer equation [79,85–87]. Dawes et al. use a lumped Tafel expression with a fixed open circuit voltage [75]. Shimpalee et al. and Meng and Wang do not include details of the reaction rate/overpotential relationship [89–91]. Um et al., Sivertsen and Djilali, and Nguyen et al. [78,86–88] use an exchange current density which depends on temperature as determined by Parthasarathy et al. [92] and used by Fuller and Newman [93].

All of the models are three-dimensional (3D) with the exception of the oldest model by Gurau et al. [94]. Most of the models in the table (2.2) are static, and this agrees with the observation of Weber and Newman over a larger set of models [68, p. 4719]. One exception is the work by Wang and Wang, but even it assumes constant temperature [79,86]. About half of the models are isothermal and half consider liquid water. Of those that do include liquid water, several (at least Um et al. [87], Shimpalee et al. [89], and Obayopo et al. [95]) assume that the liquid and gas phases have the same velocity.

Most of the physics-based models are implemented and simulated using a CFD package. The CFD software spatially discretizes the physical domain so that a particular numerical method (often FVM) can be applied to convert the problem into a system of algebraic equations (if

static) or differential algebraic equations (if dynamic). This class of PEMFC models is growing rapidly due to the recent advancements in computing power and CFD packages. In fact, three major CFD companies (ANSYS, CD-adapco, COMSOL) offer specialized off-the-shelf modules for fuel cell simulation [96–98]. These have been used in at least one of the models listed in Table 2.2 (Shimpalee et al.).

Due to the computational expense of CFD simulations, physics-based PEMFC models are usually limited to the single-cell or even sub-cell level (e.g., GDL). The model of Shimpalee et al. is one exception; it encompasses an entire stack [89].

Table 2.2: Features of selected physics-based PEMFC models. The bold entries indicate the most detailed modeling choices.

Author(s) and citation(s)	Year	Dimen- sions	Dy- namic	Liquid	Iso- thermal	Comp- ressible	Material diffusion
Gurau et al. [94]	1998	2	No	Yes	No	No	Fick's law, dependent coefficients
Dutta et al. [84]	2000	3	No	No	Yes	No	Fick's law, dependent coefficients
Um et al. [86, 87]	2000	3	No	Yes	Yes	No	Unknown method
Berming & Djilali [77]	2003	3	No	Yes	No	Yes	Maxwell-Stefan
Mazumder & Cole [81, 82]	2003	3	No	Yes	No	No	Maxwell-Stefan
Nguyen et al. [78]	2004	3	No	No	No	Yes	Maxwell-Stefan
Meng & Wang [91]	2005	3	No	No	Yes	No	Fick's law
Sivertsen & Djilali [88]	2005	3	No	Yes	No	No	Maxwell-Stefan
Wang & Wang [79]	2006	3	Yes	No	Yes	No	Fick's law
Dawes et al. [75]	2009	3	No	No	Yes	No	Fick's law, dependent coefficients
Shimpalee et al. [89]	2009	3	No	Yes	No	Unknown	Unknown method
Chippar & Ju [85]	2012	3	No	No	No	No	Fick's law, dependent coefficients
Sousa et al. [83]	2012	3	No	Yes	Yes	No	Maxwell-Stefan
Obayopo et al. [95]	2013	3	No	Yes	No	No	Fick's law

2.3.2 Phenomenological

Phenomenological or semi-empirical PEMFC models use correlations rather than momentum balances (i.e., Navier-Stokes) to relate properties such as pressure to material transport rates. These correlations may be analytically derived (e.g., Poiseuille's law) or based on results from experiment or physics-based models. Most phenomenological models are computationally smaller than physics-based models, so they are of particular interest for dynamic analyses, system- or application-level simulations (e.g, a fuel cell vehicle), and real-time embedded control [99]. Phenomenological PEMFC models typically do not include as much detail as physics-based PEMFC models [73]. They are not necessarily less accurate, but the degree of uncertainty they introduce under extrapolation is generally greater.

This class includes the models of Springer et al. [64,65], Bernardi and Verbrugge [66,67], Fuller and Newman [93], Nguyen et al. [100,101], Bevers et al. [102], and many others. The models are typically written in imperative languages such as Fortran, C, or MATLAB. With the recent interest in transient behavior, there is a trend towards high-level languages and graphical, signal-based tools with built-in integration algorithms such as MATLAB/Simulink [103]. Complete fuel cell models are even commercially available for these platforms, for example MskFcStack and the fuel cell stack model in SimPowerSystems [104,105].

The phenomenological models are often based on a one-dimensional (1D) lumped-parameter approach through the layers of the cell. They have, however, been developed with up to two or even three dimensions in a limited manner (i.e, quasi-3D) [106]. The spatial discretization is manual and much lower in resolution than the physics-based models. For instance, the most complex phenomenological model, that of Park and Min, has 225 control volumes [106], whereas the physics-based models of Shimpalee et al. and Wang and Wang have 4,823,906 and 100,000 grid points, respectively [79,89].

Franco et al. have developed a multi-scale, modular membrane electrode assembly (MEA) model based on irreversible thermodynamics and electrostatics. It is able to predict the effects of nominal current, reactant pressures, cell temperature, and electrode composition on the electro-impedance spectra. Franco et al. used bond graphs to determine the appropriate

causality at the nano scale. The model does not encompass the entire cell. It assumes that the feed gases are pure (no nitrogen) and saturated with water [107, 108].

Some phenomenological PEMFC models have been developed specifically for control system analysis and design. The group of Stefanopoulou and Peng have been particularly active in the area of modeling for PEMFC controls. Of this group, Pukrushpan has addressed a broad range of topics within fuel cell control (e.g., real-time observation, multi-variable control, and air management) [109]. Vahidi has applied model predictive control (MPC) to PEMFC systems [110, 111]. The group has also created a detailed water dynamics model to support water management and a model with nitrogen accumulation in the anode to optimize the purge cycle of a PEMFC without anode recirculation [112, 113].

2.3.3 Declarative

Declarative or equation-based fuel cell models support symbolic manipulation. As mentioned in Section 1.1.1, this allows a modeling tool to solve a model for the desired causality, linearize it, perform index reduction, and improve the numerical efficiency of simulations. In contrast, most multidimensional fuel cell models have fixed causality [91]. Zenith et al. noted that many fuel cell models are not suitable for control system design because they consider current density as an input, although in reality it is determined by the interaction between the fuel cell and the external load [8]. Declarative models can in theory remove this limitation and resolve that inconsistency.

A declarative model may be modular or flat. For the purposes of the sections below, a fuel cell model is considered modular if it has interconnected sub-models that are partitioned physically below the cell level.

2.3.3.1 Modular

There are six declarative modular or equation-based, object-oriented (EEO) fuel cell models. Five are of PEMFCs and one is of a solid oxide fuel cell (SOFC). All of these models are phenomenological. They are discussed below in order of publication.

Steinmann and Treffinger developed a PEMFC model with lumped models for each of the seven layers. It uses the dusty-gas model for transport through the cathode gas diffusion layer

(GDL), which encompasses Maxwell-Stefan multi-component diffusion and Knudsen pore interactions in parallel with advective mass flow. It includes heat transport. The results exhibit activation and Ohmic losses but no concentration loss. The model does not appear to operate at open circuit. Although no details are given on the overpotential/reaction rate equation, this may indicate that the Tafel approximation is used. No model dynamics were reported [114]. Treffinger and Goedecke used the model to simulate a hybrid electric drivetrain (battery and fuel cell) [115].

Rubio et al. openly and freely shared a dynamic three-layer lumped PEMFC model which includes electro-osmotic drag and double-layer capacitance. It also includes Maxwell-Stefan multi-component diffusion and Knudsen pore interactions. The assumptions are flexible. It is capable of simulating cell flooding and electrical transients under a step electrical load. The major limitations are that it is isothermal, does not include heat generation or models of the flow plates, and does not have external fluid or thermal connections (only electrical) [116–118].

Davies and Moore published a quasi-two-dimensional (2D) (through-the-cell and along-the-channel) dynamic PEMFC model. It also includes electro-osmotic drag and has flexible assumptions, but it is isothermal. The model uses the Butler-Volmer equation by default. The catalyst layers of the published version do not include chemical transport—only reactions and charge transport [119, 120].

McCain et al. [99] implemented the model of McKay et al. [112] (mentioned above) within a declarative framework in order to perform model order reduction (MOR) and linearize the model for control studies. The model includes liquid water and considers the obstruction of pores. It focuses on one-dimensional (1D) diffusive transport through the cell layers (particularly the GDL), which is described using Fick's law via finite differences. Each species can be discretized with different resolutions. However, this requires a data bus that hides information about the interactions between species. The overpotentials and the electrical resistances are not discussed. These may not have been included in order to focus on chemical diffusion. Heat transport is neglected as well [99].

Blunier et al. created a PEMFC model in VHDL-AMS with lumped models for each layer. The model is linearly scaled to represent a fuel cell stack and simulated with system components. The model assumes that there is no pressure loss down the cathode channels, the hydrogen pressure is constant, the cell is isothermal, and there is no anode overpotential. Water can only leave the cell as vapor [121, 122].

Salogni and Colonna created a one-dimensional (1D) model of a SOFC. Along the channel, the cell is declarative, but each segment is internally causal. The model considers gas transport and storage in the channels with the assumption of ideal gas and laminar flow. It also includes the diffusion of ions through the electrolyte and heat storage in the gas and solid. The model assumes that the entire cell is adiabatic and that there is no thermal resistance in the positive electrolyte negative (PEN) unit (equivalent to the MEA in a PEMFC). The Nernst and Butler-Volmer equations are used to determine the electrochemical potential and the Maxwell-Stefan equations are used to describe multi-component diffusion. The model interfaces are compatible with the ThermoPower library described above [123].

Equivalent circuits are another type of declarative, modular representation that appears in PEMFC literature. These often encompass the electrical resistances of the flow plates, GDLs, and proton exchange membrane (PEM) as well as the electrical behavior of the electrode/electrolyte interface. The electrode/electrolyte interface may be modeled using a Randles cell (with a capacitor and resistors) or more complex representations with Warburg or constant phase elements [124]. These representations can be used to represent the key features of the impedance spectra. However, the equivalent circuits are not complete fuel cell models unless they include chemical and thermal transport and storage. These effects can be modeled using separate equivalent circuits, but it becomes difficult to integrate all the domains in a flexible manner [121]. Another drawback is that there are no analytical expressions for the parameters of the electrical elements except to consider some simple factors such as nominal current and reactant concentration [108].

2.3.3.2 Flat

Several other declarative fuel cell models are not modular below the cell or stack level. Ungethüm developed a model of the cathode side of a PEMFC system for real-time simulation, but the model does not include the electrochemistry of the cell or the anode side of the system [125]. Maringanti et al. implemented a model of the GDLs, catalyst layers, and PEM of a PEMFC in Modelica. However, it appears that the model is entirely textual with no modularity or external interfaces [126].

A line of system-level research has also developed from the PEMFC system model of Eborn et al. at the United Technologies Corporation (UTC) in 2003 [127]. The earlier work was in Modelica, but gPROMS was used in 2005 [128]. Andersson and Åberg, advised by Eborn, have also modeled SOFC systems in Modelica [129, 130].

Process Systems Enterprise Limited (PSE), the owner of the gPROMS language/environment mentioned in Section 2.1, offers gFuelCell, an off-the-shelf package for modeling PEMFCs and SOFCs. It has declarative interfaces, but it does not appear to be modular below the cell level. It supports two- and three-dimensional (3D) analyses. In some examples, the MEA is interfaced to a CFD representation of the flow plates and channels [24]. PSE claims that gFuelCell includes all the relevant physics and chemistry, but few details are publicly available [131].

Modelon AB offers a commercial fuel cell package for system simulation in Modelica. Like gFuelCell, it has declarative interfaces but does not appear to be modular below the cell level. Few details are publicly available [132].

Bruun developed a model of a SOFC system with integral causality using bond graphs. It includes heat transfer. Although the cell model consists of many bonds and junctions, the elements are not partitioned in a layer-based manner [61].

2.4 Summary

This chapter reviewed the current equation-based, object-oriented (EEO) languages, the recent work to model fluid and chemical systems using the Modelica language in particular, and the very active area of fuel cell modeling. The Modelica language was selected to implement the equations of the next chapter in a manner that is physics-based, modular, reconfigurable,

and leads to numerically efficient and robust models. The developments will be demonstrated in a fuel cell model. From the previous section (2.3), it is apparent that this will be the first declarative, physics-based fuel cell model.

FUNDAMENTALS OF THE MODEL

This chapter describes the key physical relations and behavioral equations that form the basis of the modeling approach. The thermodynamic properties are defined in a general manner that is equally applicable to ideal gases, real gases, and condensed species. The exchange, transport, and conservation equations are established at a fundamental, physics-based level. This may seem excessive, but as previously noted, it helps to provide insight into observed behavior. Since the equations are implemented flexibly using an equation-based, object-oriented (EOO) language, the modeling tool can often remove the complexity if it is not necessary. The motivating principle is that the assumptions and inaccuracies should be contained in adjustable parameters, and the complexity of the equations should scale appropriately and automatically. That way, a common modeling framework can be applied over a wider range of applications.

The disadvantage is that some of the equations are nontraditional and thus require more initial effort from the reader. To clarify how they correspond to existing theory, some well-established physical laws are derived from the model under the applicable conditions and appropriate assumptions. Where possible, the derivations are presented and discussed immediately following the model equations. However, some derivations depend on model equations from several sections; these are deferred to Appendix A. Table 3.1 lists the sections and page numbers where various physical laws and topics are discussed.

Figure 3.1 shows a *subregion*, the basic geometric block of the model. It is a control volume with fixed rectangular boundaries. All flows are positive inward. The forces, positions, and velocities are globally referenced along the x, y, and z axes. Force is considered to be the rate of momentum oriented in the globally positive direction and yet flowing into the control volume. The flows are noted in the form of \dot{X} with a subscript from the figure. For example, the subscript x_n indicates the negative boundary along the x axis.

Table 3.1: Cross-references of physical topics and laws.

Mass transfer and fluid dynamics		
3.5	Material derivative	60
3.5.1	Gibbs phase rule	61
3.7.1	Fick’s law and self diffusion	85
3.7.2.1	Dynamic pressure	88
3.7.3	Stokes’ law of viscous deformation	94
3.7.3	Newton’s law of viscosity and Couette flow	94
3.7.3	Laminar pipe flow and Poiseuille’s law	94
3.7.3	Reynolds number and turbulent flow	95
3.10.1	Continuity equation	109
3.10.2	Eulerian and Lagrangian specifications	112
3.10.2	Navier-Stokes equations	112
A.1	Darcy’s law	231
A.2	Maxwell-Stefan equations	233
A.2	Dusty-gas model	239
Heat transfer		
3.7.4	Fourier’s law	99
3.7.4	Nusselt number	98
3.7.4	Newton’s law of cooling	99
3.10.3	Heat equation	116
Electrochemistry		
3.9	Butler-Volmer equation	106
Solid-state physics		
A.5	Einstein relation	241
A.3	Charge drift/diffusion	239
A.4	Ohm’s law	240

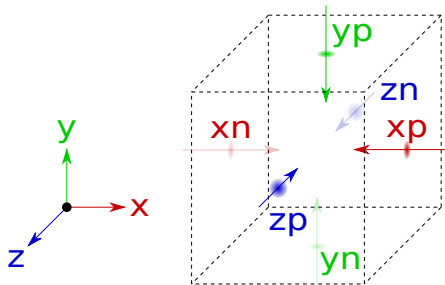


Figure 3.1: Illustration of a region or subregion.

In the implementation (Chapter 4), subregions may be connected in multiple directions to form a *region*. A region is also a rectangular cuboid. Figure 1.8 shows the region and subregion within the model hierarchy. However, this chapter only deals with one or two subregions at once. Therefore, what will be called a subregion in the model implementation is often simply called a region in this chapter.

The subregion is the lowest level of spatial resolution, but it may contain multiple *species* in multiple *phases*. Each of these *configurations*, or species in certain phases, are treated as distinct but interacting entities. The species and phases are also control volumes, but the model does not directly resolve the shape, location, and orientation of their boundaries.

Material, momentum, or energy may be *transferred* among control volumes. *Material* is synonymous with matter; it represents particles, atoms, or molecules. Material is distinct from momentum and energy, although the transfer of material generally carries momentum and energy. Since the model deals with chemical reactions and phase change, material is measured in terms of a number (which may be expressed in a unit such as the mole) rather than mass. Material is also used as an adjective, as in material transfer. *Current* is the flow rate of material, which may or may not be charged. Electrical current is the flow rate of charge. These and other key terms are listed in the glossary on page xxiii.

There are two types of transfer. *Exchange* is transfer among different configurations within a subregion. *Transport* is transfer between similar configurations in neighboring subregions. Thus, exchange is local and transport is spatial in nature.¹ Chemical reactions and phases change involve material exchange—that is, transfer of matter among various configurations

¹Microscopically, exchange is also spatial, but the model is macroscopic.

governed by the laws of chemistry. Both types of transfer (exchange and transport) can generally occur by advection or diffusion. These processes were introduced in Section 1.1.2 and will be discussed further in this chapter.

So far, two assumptions have been introduced: (1) all of the boundaries are rectangular and (2) the subregions and regions have fixed boundaries. Further assumptions are introduced as necessary throughout the chapter.

It is important to note that the model equations are presented for a unit system where the gas and Faraday constants are normalized to one (see Section 4.3). This simplifies the equations and their implementation. Also note that the *specific* adjective is used to indicate “per unit amount of material”.ⁱⁱ For example, specific mass is the mass per unit number of particles. It is used instead of molar mass because the unit system is neutral with respect to the unit that represents the amount of material.

Figure 3.2 shows the high-level aspects that must be considered in a fuel cell model. These will each be discussed in the following sections, with the exception of geometry (see above). Many of the sections begin with a list of key features of the model that are either unusual or are new contributions. The boxed equations are the ones that are actually implemented in the model (next chapter). Other equations are presented to provide insight into the implemented equations and relate the model to established theories.

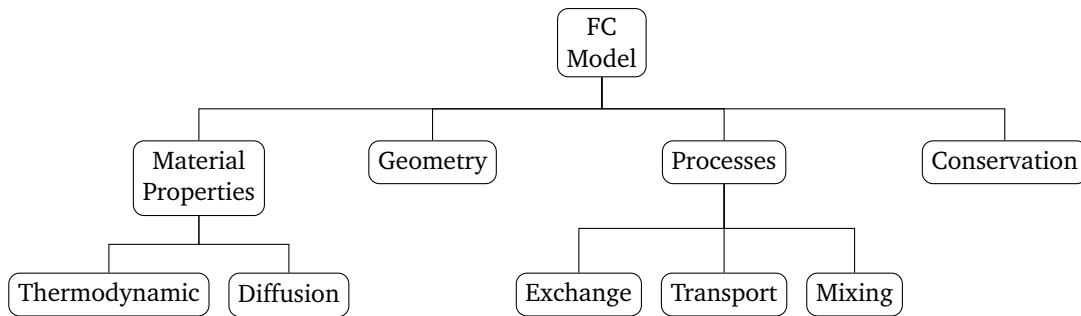


Figure 3.2: Considerations of a fuel cell model.

ⁱⁱIn contrast, massic is defined by the International Standards Organization (ISO) to mean “per mass” [133].

3.1 Correlated Thermodynamic Properties

Highlights:

- The correlations are necessary and sufficient to calculate basic thermodynamic properties (specific entropy, enthalpy, Gibbs energy, etc.) given temperature and pressure or specific volume.
- The correlations are polynomial and can be expanded as needed for accuracy.
- The pressure-volume-temperature correlation is sufficiently general to describe ideal gases, real gases, and incompressible species with or without thermal expansion. Many fuel cell models are explicitly based on ideal gases under incompressible flow [64, 66, 79, 86–88, 100, 101, 118, 134–141].
- The specific heat capacity-temperature correlation is general enough to model media with constant specific heat or to provide accurate information on the temperature dependence. This makes it possible to simplify the descriptions of inert gases (e.g., N₂) and more accurately describe reacting gases (H₂, H₂O, and O₂) to model the temperature dependence of the cell potential.

3.1.1 Isobaric Specific Heat Capacity-Temperature Relation

Isobaric specific heat capacity is defined by

$$c_p \equiv T \left(\frac{\partial s}{\partial T} \right)_p \quad (3.1)$$

where s is specific entropy, T is temperature, and p is pressure. McBride et al. provide the isobaric specific heat capacity of many species as a correlated polynomial of temperature [142]; however, c_p is in general a function of both temperature and pressure. For condensed species, they specify the thermodynamic state by the actual temperature and a reference pressure (1 atm). The state of gases is chosen to be the ideal gas at the given temperature, since the

specific heat capacity of an ideal gas is independent of pressure. The correlation for this adjusted isobaric specific heat capacity is

$$c_p^o = b_1 T^{-2} + b_2 T^{-1} + b_3 + b_4 T + b_5 T^2 + b_6 T^3 + b_7 T^4 \quad (3.2)$$

The coefficients (b_1, b_2, \dots) must be chosen for the proper temperature range but are otherwise constant. The model uses a more general form

$$c_p^o = \sum_{i=1}^m b_i T^{i+n-1} \quad (3.3)$$

where n is the power of the first term and the polynomial has an arbitrary number of terms (m). The order of the polynomial is $m + n - 1$. Multiple sets of coefficients may be specified; they are selected depending on the temperature range (as per McBride et al.).

3.1.2 Pressure-Volume-Temperature Relation

The model uses the virial equation of state (EOS), which was proposed by Thiesen in 1885 and validated against many gases by Kammerling-Onnes in 1901 [143,144]. It is convenient for use with differential equations and can be expanded as needed for accuracy. The virial EOS can be expressed in a volume-explicit (or Leiden [145]) form as

$$v = b_1 \left(\frac{p}{T}\right)^{-1} + b_2 + b_3 \left(\frac{p}{T}\right)^1 + b_4 \left(\frac{p}{T}\right)^2 + \dots \quad (3.4)$$

where the coefficients are functions of temperature only [146]. The model uses the following generalized form

$$v = \sum_{i=1}^{m_1} \sum_{j=1}^{m_2} b_{ij} \left(\frac{p}{T}\right)^{i+n_1-1} T^{j+n_2-1} \quad (3.5)$$

where n_1 and n_2 are the powers of the first term and the polynomial has an arbitrary numbers of terms in both dimensions (m_1, m_2). The p/T group is used instead of p so that (1) the matrix of coefficients (b_{ij}) is more compact for typical correlations (e.g., [146]) and (2) the virial inverse matrix (b'_{ij} below) has the same size.

The virial coefficients may be derived from the statistical mechanics of intermolecular forces [147]. For gases, the first virial coefficient (b_1) is generally the gas constant, which

has been normalized to one. The second virial coefficient characterizes binary interactions between molecules—specifically the pair energy potential function [146]. The third coefficient is for ternary interactions, the fourth is for quaternary interactions, and so on [145]. These effects diminish rapidly with the order of the interaction [148]. If $b_1 = 1$ and the other terms are neglected, the virial EOS reduces to the ideal gas EOS. For gases at low pressures, only the first and possibly the second virial coefficients are necessary. Dymond et al. correlate the second virial coefficients of many gases to polynomials in temperature [146].

Equation 3.4 is suitable for incompressible or even constant-volume species, where only the second virial coefficient (b_2) is nonzero. If the species is compressible, the volume-explicit form of Equation 3.4 is equivalent to the following pressure-explicit (or Berlin) form [145, 146]:

$$\frac{P}{T} = b'_1 v^{-1} + b'_2 v^{-2} + b'_3 v^{-3} + b'_4 v^{-4} + \dots \quad (3.6)$$

Otherwise, pressure cannot be determined from temperature and specific volume. The model uses the following generalized form:

$$v = \sum_{i=1-m_1}^0 \sum_{j=1}^{m_2} b'_{ij} P^{i+n_1} T^{j+n_2} \quad (3.7)$$

The modified coefficients of Equation 3.6 are directly related to those of Equation 3.4 [145, 146]:

$$b'_1 = b_1 \quad (3.8a)$$

$$b'_2 = b_2 \quad (3.8b)$$

$$b'_3 = b_2^2 + b_3 \quad (3.8c)$$

$$b'_4 = b_2^3 + 3b_2 b_3 + b_4 \quad (3.8d)$$

We can determine the relations for even higher-order coefficients by setting Equations 3.4 and 3.6 equal (in terms of pv/T) and successively eliminating terms [147].

3.2 Derived Thermodynamic Properties

Highlights:

- The derivations are exact and do not involve additional assumptions besides those inherent in the correlated properties of Section 3.1.
- The properties are general and complete enough that the model does not require specialized thermodynamic correlations such as the saturation pressure-temperature curve of water (H₂O).

3.2.1 Specific Entropy

We can write specific entropy as

$$s = \int_0^T \frac{c_p^o}{T} dT + \int_{p^o}^p \left(\frac{\partial s}{\partial p} \right)_T dp \quad (3.9)$$

where c_p^o is evaluated at reference pressure p^o . Applying the appropriate Maxwell relation, $(\partial s / \partial p)_T = -(\partial v / \partial T)_p$, this is

$$s = \int_0^T \frac{c_p^o}{T} dT - \int_{p^o}^p \left(\frac{\partial v}{\partial T} \right)_p dp \quad (3.10)$$

which can be evaluated using Equations 3.2 and 3.4. McBride et al. [142] give the integration constant of the first term for each species so that the isobaric specific heat correlation does not need to be evaluated at (or even valid at) absolute zero temperature. The second integral is $\ln(p/p^o)$ for an ideal gas and typically small for condensed species. Again, the second- and higher-order virial coefficients (b_2, b_3, \dots) are functions of temperature but not pressure. The coefficients of isobaric specific heat capacity (b_1, b_2, \dots) may be treated as constant but must be chosen based on the temperature range.

For gases, the lower limit of the second integral of Equation 3.10 is evaluated only for the first virial coefficient. This adjustment is necessary because the reference for the c_p - T correlation is the ideal gas instead of the real gas. Formally, the modified form of Equation 3.10 for gases is

$$s = \int_0^T \frac{c_p^o}{T} dT - \left[\int_{p^o}^0 \left(\frac{\partial v_{IG}}{\partial T} \right)_p dp + \int_0^p \left(\frac{\partial v}{\partial T} \right)_p dp \right] \quad (3.11)$$

where the first integral in square brackets involves the specific volume of the ideal gas and the second involves the real gas. Following the approach by Rao [149], the ideal gas contribution

is integrated from the reference pressure to zero pressure and the real gas contribution is integrated from zero pressure to the actual pressure. At zero pressure, binary and higher-order molecular interactions are eliminated and a real gas behaves as an ideal gas.

Figure 3.3 depicts the integration path of the pressure terms. Assuming that the first virial coefficient (b_1) is the same for the ideal gas and the real gas (since for gases $b_1 = 1$), the contribution of the first-order virial term may be integrated directly from p° to p . In effect, this combines $\ln(0/p^\circ) + \ln(p/0)$ to give $\ln(p/p^\circ)$. Since the contributions of the second- and higher-order virial coefficients are zero at zero pressure, we can eliminate those integral evaluations. The net result is that the lower limit of the second integral in Equation 3.10 is evaluated for the ideal gas and the upper limit is evaluated for the real gas.

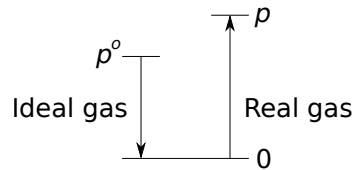


Figure 3.3: Integration path for the specific entropy of gases.

3.2.2 Specific Enthalpy

We can define specific enthalpy by the differential equation

$$dh = Tds + vdp \quad (3.12)$$

After substituting Equation 3.10 and integrating with an adjustable temperature reference, this is

$$h = \int_{T^\circ}^T c_p^\circ dT + \int_{p^\circ}^p \left[v - T \left(\frac{\partial v}{\partial T} \right)_p \right] dp \quad (3.13)$$

which can be evaluated using Equations 3.2 and 3.4. As for specific entropy, if the species is a gas, the lower limit of the second integral is of the ideal gas and the upper limit is of the real gas. The second integral is zero for an ideal gas and typically small for condensed species. McBride et al. [142] give the sufficient integration constants and offsets to specify the enthalpy reference such that (1) the enthalpy at 0 K and p° is zero, (2) the enthalpy at 25 °C and p° is zero, or (3) the enthalpy at 25 °C and p° is the enthalpy of formation at that temperature and pressure.

3.2.3 Specific Gibbs Energy

We can define specific Gibbs energy by the following differential equation:

$$dg = vdp - sdT \quad (3.14)$$

In conjunction with Equation 3.12, this implies that

$$g = h - Ts \quad (3.15)$$

Substituting Equations 3.10 and 3.13,

$$g = \int_{p^o}^p v dp + T \int_{p^o}^p \left(\frac{\partial v}{\partial T} \right)_p dp - \int_{p^o}^p T \left(\frac{\partial v}{\partial T} \right)_p dp + \int_{T^o}^T c_p^o dT - T \int_0^T \frac{c_p^o}{T} dT \quad (3.16)$$

which can be evaluated using Equations 3.2, 3.4, and 3.6. If the species is a gas, the lower limits of the pressure integrals are of the ideal gas and the upper limits are of the real gas (see Section 3.2.1).

3.2.4 Isobaric Specific Heat Capacity

We can express the isobaric specific heat capacity by substituting Equation 3.10 into Equation 3.1.

$$c_p = c_p^o - T \left(\frac{\partial \left[\int_{p^o}^p \left(\frac{\partial v}{\partial T} \right)_p dp \right]}{\partial T} \right)_p \quad (3.17)$$

This can also be evaluated using Equations 3.2 and 3.4. The second term is zero for an ideal gas and usually small for condensed species. If the species is a gas, the lower limit of the integral is of the ideal gas and the upper limit is of the real gas (see Section 3.2.1).

3.2.5 Isochoric Specific Heat Capacity

Isochoric specific heat capacity is defined by

$$c_v \equiv T \left(\frac{\partial s}{\partial T} \right)_v \quad (3.18)$$

The isochoric and isobaric specific heat capacities are related by the following equation [150]:

$$c_v = c_p - T \left(\frac{\partial p}{\partial T} \right)_v \left(\frac{\partial v}{\partial T} \right)_p \quad (3.19)$$

Applying Equation 3.17, this is

$$c_v = c_p^o - T \left[\left(\frac{\partial p}{\partial T} \right)_v \left(\frac{\partial v}{\partial T} \right)_p + \left(\frac{\partial \left[\int_{p^o}^p \left(\frac{\partial v}{\partial T} \right)_p dp \right]}{\partial T} \right)_p \right] \quad (3.20)$$

which can be evaluated using Equations 3.2, 3.4, and 3.6. It reduces to $c_v = c_p - 1$ for an ideal gas. If the species is a gas, the lower limit of the integral is of the ideal gas and the upper limit is of the real gas (see Section 3.2.1).

3.3 Mixtures

Highlights:

- Traditionally, Dalton's and Amagat's laws are used with the ideal gas assumption, but the model does not impose that requirement.ⁱⁱⁱ
- The volumes and pressures of mixtures can change dynamically, but the model imposes Dalton's and Amagat's laws exactly and instantaneously. There are no additional states.

3.3.1 Species within a Phase

The model combines species within a phase using Dalton's law of partial pressures, which states that the partial pressures of the components of a mixture add to the total pressure of the mixture [151]:

$$p = \sum p_i \quad (3.21)$$

Dalton's law also states that each species i exists at the total volume of the phase:

$$V_i = V \quad (3.22)$$

For example, according to this concept, the atmospheric gases of N_2 , O_2 , etc. each occupy the total volume of the air but only contribute partially to the pressure of the air.

3.3.2 Phases within a Region

The model combines phases within a region using Amagat’s law of partial volumes, which states that the partial extensive volumes of the components of a mixture sum to the total extensive volume of the mixture [151]:

$$V = \sum V_i \quad (3.23)$$

In the model, V_i is the volume of a phase and V is the volume of the region, which is fixed. Amagat’s law also states that each species i exists at the total pressure of the phase:

$$p_i = p \quad (3.24)$$

The model only uses Amagat’s law for distinct phases within a region—not for species within a phase. Amagat’s law loses its physical meaning as species are mixed [152]. If species are fully mixed, it is impossible to distinguish the particles and thus determine the partial volumes.

For example, if a system contains a solid phase and air, the model states that the solid and the air experience the same pressure and occupy only part of the total volume (Amagat’s law). Within the air, the gases mix according to Dalton’s law (Section 3.3.1). The model applies Dalton’s law and Amagat’s law dynamically, which makes it possible to describe the formation of liquid water in the cell [68, 102, 153].

The model is classified as a Euler-Euler approach rather than a Euler-Lagrange approach [154], since all phases are tracked from a Eulerian perspective. The volume fractions are continuous functions of time and must sum to one. The Euler-Lagrange approach is limited to problems where the solid phase has a small volume in comparison to the fluid phase—10% to 12% [154]. This is not appropriate for the layers of a fuel cell.

3.4 Basic Conservation Equations

Highlights:

- The model is dynamic. It includes material, momentum, and energy storage.
- Each species has its own conservation equations for material, translational momentum, and energy. However, the model’s parameters can be set so that the translation

tool combines certain conservation equations through index reduction. The concept of separate momentum balances for each species is unusual but not unprecedented in the literature [155–157]. Separate energy balances are rarely used ([158] is one example).

Material, momentum, and energy are conserved throughout the model at interfaces and within regions. Each configuration (i.e., each species in each phase) has its own conservation equation in every region. The conserved quantities can be stored in configurations but not at interfaces between or among configurations.

Below, the conservation equations (i.e., balances) are introduced with minimal detail to explain the exchange and transport equations (Sections 3.5 and 3.7). The interfaces are generalized here, but there are two types: boundaries between regions (for transport) and transitions among configurations within a region (for exchange). In general, the flow through each interface has advective and diffusive components. Later, in Section 3.10, detailed conservation equations will be presented.

3.4.1 Material

The rate of storage of material is equal to the net rate of intake or transfer of material into a control volume. Figure 3.4 shows that there are two types of material transfer—exchange and transport. In general, exchange and transport can each occur by advection or diffusion. However, the model considers chemical reactions and phase change, the two modes of material exchange, to be diffusive processes.

For now, the material balance will be written simply as

$$\underbrace{\frac{\partial N}{\partial t}}_{\text{storage}} = \underbrace{\sum \dot{N}_i}_{\text{intake}} \quad (3.25)$$

where N is the particle number or amount of material and \dot{N}_i is the total current (advective and diffusive, $\dot{N}_{A_i} + \dot{N}_{D_i}$) into a generalized interface i . The use of a partial derivative ($\frac{\partial N}{\partial t}$) rather than a total derivative ($\frac{dN}{dt}$) serves as a reminder that the model is Eulerian. Although

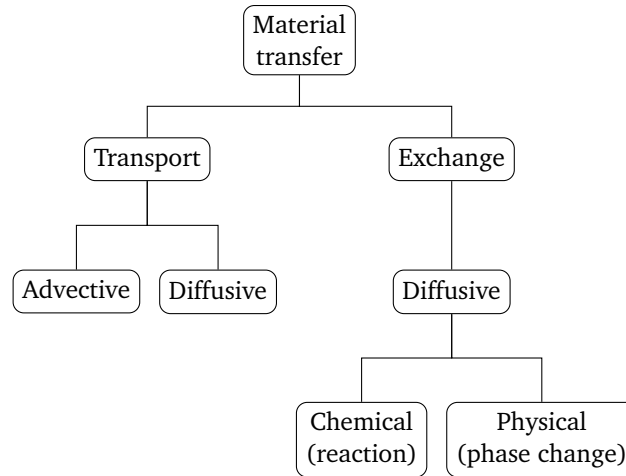


Figure 3.4: Types of material intake considered in the model.

the equation is written in terms of material, mass is conserved as well since the specific mass of each configuration is constant and the phase change and reaction processes are balanced in terms of mass.

At boundaries and transitions, there is no material storage. Advection has no net effect because the rate of advection is continuous across a boundary. Therefore, the material balance reduces to

$$0 = \sum \dot{N}_{Di} \quad (3.26)$$

where the summation is now across all interacting configurations i . This equation is generated automatically by the connection equations of the EOO language; it is the generalized Kirchhoff current law for material.

3.4.2 Rotational Momentum

The model is based on the assumption that rotational momentum is not stored. Rotational momentum is not exchanged or transported axially through boundaries, but it is conveyed through shear forces. We assume that the forces are point forces in the center of the boundaries

and that the axes of rotation are centered in the region. Therefore,

$$0 = \left(\dot{m}\Phi_{zny} - \dot{m}\Phi_{zpy} \right) L_z - \left(\dot{m}\Phi_{ynz} - \dot{m}\Phi_{ypz} \right) L_y \quad (3.27a)$$

$$0 = \left(\dot{m}\Phi_{xnz} - \dot{m}\Phi_{xpz} \right) L_x - \left(\dot{m}\Phi_{znx} - \dot{m}\Phi_{zpx} \right) L_z \quad (3.27b)$$

$$0 = \left(\dot{m}\Phi_{ynx} - \dot{m}\Phi_{ypx} \right) L_y - \left(\dot{m}\Phi_{xny} - \dot{m}\Phi_{xpy} \right) L_x \quad (3.27c)$$

where $\dot{m}\Phi_{zny}$ is the force in the y-direction through the negative-z boundary, $\dot{m}\Phi_{ypz}$ is the force in the z direction through the positive-y boundary, and so on. The normal forces do not introduce torque since they are aligned with the center of rotation. These equations are included in the diffusion equations for shear force around each axis (Section 3.7.3).

3.4.3 Translational Momentum

The rate of storage of translational momentum is equal to the sum of the forces on a control volume. As shown by Figure 3.4, there are three types of forces on a configuration within a region: body forces, surface forces, and intermolecular forces. The body forces may be gravitational or electric; magnetic and nuclear forces are assumed to be negligible. The surface forces include the effects of thermodynamic pressure, advection (i.e., dynamic pressure), and diffusion (i.e., nonequilibrium pressure [159] and shear stress). The thermodynamic pressure is always normal to the surface, but advection and diffusion also have transverse components. The intermolecular or exchange forces may be advective or diffusive. Advective exchange occurs, for example, in a reacting stream where the reactants are traveling relative to the control volume. Diffusive exchange occurs in multi-component fluids when the species are traveling at different velocities.

For now, we will generalize the advective and diffusive forces to encompass both exchange and transport. We will also combine the normal and transverse components of transport. Therefore, the translational momentum balance can be written as

$$\underbrace{\frac{\partial (M\phi)}{\partial t}}_{\text{storage}} + \underbrace{Ma + ZE}_{\text{body}} + \underbrace{A\Delta p_i}_{\text{thermo-dynamic}} = \sum \left(\underbrace{m\phi_i \dot{N}_i}_{\text{advection}} + \underbrace{\dot{m}\Phi_{Di}}_{\text{diffusion}} \right) \quad (3.28)$$

where ϕ is a component of velocity, E is the electric field, and a represents the acceleration due to additional body forces. The mass M is mN and the charge Z is zN , where m is the

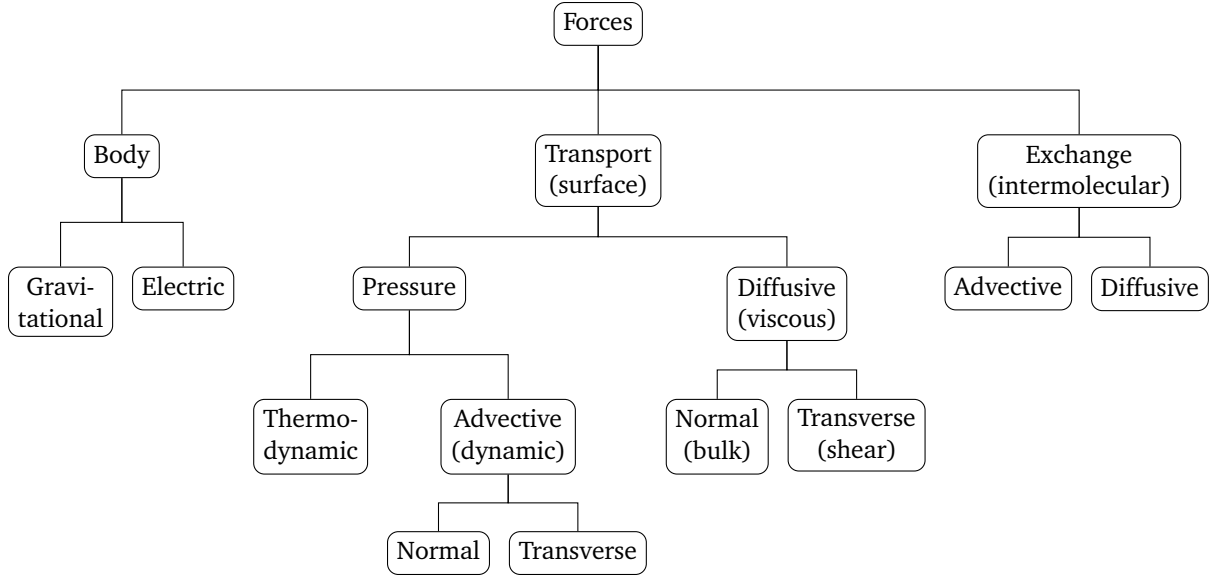


Figure 3.5: Types of forces considered in the model.

specific mass, z is the charge number, and N is the amount of material known from the state of the material balance (3.25). The diffusive terms ($m\dot{\Phi}_D$) include shear forces, nonequilibrium normal forces, and drag among configurations.

The advective forces ($m\phi\dot{N}$) account for convective acceleration and the momentum transferred in reacting flows and phase change. It is important to note again that the material transfer (\dot{N}) includes both advection and diffusion. Momentum is advected or carried by material regardless of whether that material is transferred by advection or diffusion.

The difference (Δ) on the left side of Equation 3.28 is across the boundaries normal to the component of translational momentum. The variable A is the area of those boundaries. The thermodynamic force term ($A\Delta p_i$) is based on the assumption that the configuration experiences pressure across the entire cross-sectional area of the region, although in reality the area is reduced if other phases are present. This is necessary to ensure that translational momentum is conserved between two adjacent regions.

At boundaries and transitions, there is no storage. The advective terms of Equation 3.28 cancel because the advected properties are continuous at the interface. The thermodynamic pressure is continuous at the interface and thus has no effect. There is no material in the interface, so there are no body forces. Therefore, the conservation of translational momentum

reduces to

$$0 = \sum m\Phi_{Di} \quad (3.29)$$

where the summation is now across all interacting configurations i . This equation is generated automatically by the connection equations of the EOO language; it is the generalized Kirchhoff current law for translational momentum.

3.4.4 Energy

The rate of storage of energy in a control volume is equal to the net rate of intake. As shown in Figure 3.6, the intake can be divided into material, translational, and thermal parts. Although not shown, each of these forms may be transferred by exchange or transport. The translational and thermal energy transfers can be due to advection or diffusion. The material transfer of energy is not labeled as advective or diffusive in Figure 3.6 because it requires further explanation. The material itself can be transferred by advection or diffusion, but the associated energy transfer is purely advective because the energy is carried by material.

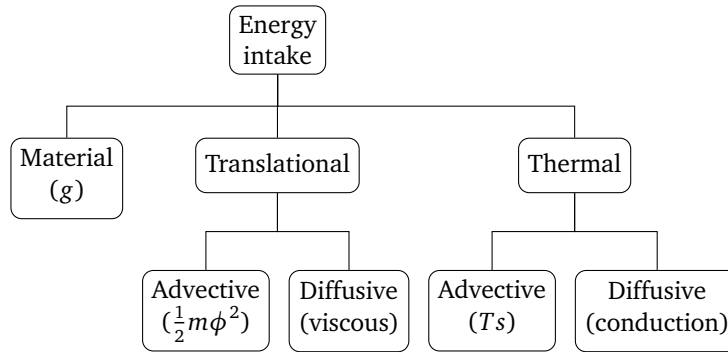


Figure 3.6: Types of energy intake considered in the model.

Thermal conduction is synonymous with thermal diffusion. Thermal convection is the combined effect of diffusive thermal exchange between phases (often a solid and a fluid) and the subsequent transport of thermal energy via advection of the fluid. The thermal advection factor (Ts) and the material factor (g) constitute specific enthalpy ($h = g + Ts$, Equation 3.15). Combined with the translational advection factor, this is specific enthalpy plus specific kinetic energy ($h + \frac{1}{2}m\phi^2$).

The factors of Figure 3.6 are incorporated into the right side of the energy balance equation:

$$\underbrace{\underbrace{g \frac{\partial N}{\partial t}}_{\text{material}} + \underbrace{\frac{\partial (M\phi^2)}{2\partial t}}_{\text{translational}} + \underbrace{T \frac{\partial S}{\partial t}}_{\text{thermal}}}_{\text{storage}} = \underbrace{\sum \left[\underbrace{g_i \dot{N}_i}_{\text{material}} + \underbrace{\phi_i \left(\frac{m\phi_i}{2} \dot{N}_i + m\Phi_{Di} \right)}_{\substack{\text{advection} \quad \text{diffusion} \\ \text{translational}}} + \underbrace{(Ts)_i \dot{N}_i + \dot{Q}_{Di}}_{\substack{\text{advection} \quad \text{diffusion} \\ \text{thermal}}} \right]}_{\text{intake}} \quad (3.30)$$

where \dot{N}_i is the total current (advective and diffusive) into interface i . This equation applies to every species in every phase. It has been assumed that the control volume is stationary with respect to external fields (e.g., no gravitational work), although the fluid may move against those fields within the control volume. The translational diffusion term, in conjunction with the translational (or kinetic) storage term, accounts for viscous dissipation. This will be more apparent in later forms of the energy balance (e.g., Equation 3.201).

The material and thermal storage terms ($g \frac{\partial N}{\partial t} + T \frac{\partial S}{\partial t}$) are equivalent to $\frac{\partial H}{\partial t} - V \frac{\partial p}{\partial t}$ or $\frac{\partial U}{\partial t} + p \frac{\partial V}{\partial t}$, where $p \frac{\partial V}{\partial t}$ is the boundary work done by the configuration.^{iv} The boundary work can only be due to expansion of the phases in which the configuration belongs (and contraction of other phases) because the volume of the region is fixed. The translational storage term describes the change in macroscopic kinetic energy.

At boundaries and transitions, there is no energy storage. The advective terms cancel because the advected properties are continuous at the interface. The translational diffusion terms can be removed because they sum to zero according to Equation 3.29. Therefore, the conservation of energy reduces to

$$\boxed{0 = \sum \dot{Q}_{Di}} \quad (3.31)$$

where the summation is now across all interacting configurations i . This equation is generated automatically by the connection equations of the EOO language; it is the generalized Kirchhoff current law for heat transfer.

^{iv}The form of Equation 3.30 has been chosen so that the material, translational, and thermal terms are explicit on both sides. Specific flow work ($p\nu$) is considered a part of the material term.

3.5 Exchange Equations

Highlights:

- A common modeling framework is used for phase change, intermolecular drag, and intermolecular thermal conduction.
- The transfer of translational momentum and energy due to phase change and reactions is described as pure advection.
- An analogy is established between the total (advective plus diffusive) rate of exchange and the material derivative.
- The model describes phase change dynamically. It does not assume instantaneous phase equilibrium in the sense of the Gibbs phase rule [150, 151]. This avoids nonlinear systems of equations while using the previously established properties (i.e., no need to establish a separate correlation for saturation pressure).
- The rate of phase change is proportional to the difference in chemical activity between the phases.
- A property called independity is defined which generalizes the concept of mobility for translational interactions to thermal interactions.

Exchange is the transfer of a conserved quantity—material, momentum, or energy—among different configurations of material that exist within a region. In general, it is due to advection and diffusion. Advective exchange is the transfer of the quantity along with a sustained transfer of material between species (i.e., reaction) or different phases of a single species (i.e., phase change). Diffusive exchange is the transfer of the quantity due only to collisions or thermal agitation of the particles, without a sustained material transfer. In diffusion, a particle leaves one configuration (i.e., a species in a certain phase) with the specific quantity (or particle-average amount of the quantity) within the configuration and returns with the specific quantity of the other configuration. This brings certain intensive properties—specific Gibbs energy, velocity, and temperature—into equilibrium among the configurations.

The model of exchange is based on the assumption that advection and diffusion are independent yet additive. The total rate of exchange of the quantity X into a configuration j due to interaction or transition i is the sum of the advective and diffusive rates:

$$\dot{X}_{ij} = \dot{X}_{Aij} + \dot{X}_{Dij} \quad (3.32)$$

For material exchange, the quantity (X) is the amount of material (N). For translational exchange, it is the product of the amount of material and velocity (Φ). For thermal exchange, it is heat (Q). The phase change and reaction processes are purely advective in terms of translational momentum and energy. Particles from the source (e.g., reactants) carry properties through the process without intermediately mixing with particles from the sink (e.g., products). Meanwhile, there are diffusive interactions for translational momentum and energy that are independent of the phase change and reactions.

Advection

The rate of advective exchange is the product of the current and the amount of the exchanged quantity carried by the material.

$$\dot{X}_{Aij} = \dot{N}_{ij} \left(\frac{\partial X}{\partial N} \right)_{ij} \quad (3.33)$$

The variable \dot{N}_{ij} is the rate of material exchange. Both \dot{N}_{ij} and \dot{X}_{Aij} are due to the interaction (i) and are directed into the configuration (j).

The partial derivative, $\partial X/N$, is an intensive property. For material advection, it is unity (1);^v for translational exchange, it is velocity (ϕ); and for thermal exchange, it is the product of specific entropy and temperature (sT). Since advection and diffusion are independent, there is no intermediate mixing between the sources and sinks. Therefore, the upwind scheme is appropriate. Here, it is applied locally among configurations rather than spatially between regions. Using the upwind scheme, the previous equation (3.33) can be written as

$$\dot{X}_{Aij} = \dot{N}_{ij} \cdot \begin{cases} \left(\frac{\partial X}{\partial N} \right)_j & \text{if } \dot{N}_{ij} < 0 \text{ (source),} \\ \left(\frac{\partial X}{\partial N} \right)_i & \text{if } \dot{N}_{ij} > 0 \text{ (sink)} \end{cases} \quad (3.34)$$

^vIn this case Equation 3.33 reduces to an identity and is removed from the model. This is consistent with the previous statement (Section 3.4.1) that the only mode of material exchange is diffusion.

The factor $\left(\frac{\partial X}{\partial N}\right)_i$ is called the *conversion property* because it is the property at which the sources (e.g., reactants) are converted to the sinks (e.g., products). The designations of source and sink depend on the direction of the phase change or reaction at a given time.

Diffusion

We can consider the rate of diffusive exchange to be the material derivative or the rate of transfer experienced by the particles themselves.^{vi}

$$\dot{X}_{Dij} = \left(\frac{DX}{Dt}\right)_{ij} \quad (3.35)$$

$(Dt)_{ij}$ is the product of the mean collision interval (τ_j) between particles and $8/3\pi$ as a result of the Einstein relation. We will assume that the exchanged quantity is linear with respect to an intensive driving property γ ; therefore, $(DX)_{ij} = (\gamma_i - \gamma_j)(\partial X / \partial \gamma)_j$. The following assumptions have also been implied:

1. The collision events are frequent enough for the average collision interval to be meaningful. This implies that the mean free path, or the average distance traveled between collisions, is much smaller than the length scale of the problem. It is not the case for example in effusion [148].
2. Between collisions the particles have no influence on one another.
3. The properties of a particle depend only on those of the last particle with which it collided.

In practice, these assumptions may be relaxed by using empirical diffusion coefficients (see Section 3.6). It follows that

$$\frac{8}{3\pi} \tau_j \dot{X}_{Dij} = k_{ij} \left(\frac{\partial X}{\partial \gamma}\right)_j (\gamma_i - \gamma_j) \quad (3.36)$$

where the dimensionless adjustment factor k_{ij} has been introduced to account for the effect of geometry (e.g., one gas species will typically be coupled more strongly to another than to a solid species) and to add the degrees of freedom necessary to match an arbitrary set of Maxwell-Stefan binary diffusion coefficients (see Section A.2). It is one by default. If two or more interacting configurations have collision intervals of zero, their driving properties (γ_j)

^{vi}See the discussion on 60.

will be equal. The transported quantity will be exchanged without loss and the number of dynamic states will be reduced. The partial derivative $(\partial X/\partial \gamma)_j$ is an extensive property of the configuration. For material exchange, it is volume; for translational exchange, it is amount of material; and for thermal exchange, it is heat capacity. The variable γ_i is called the *mediation property* because it is the property to which the differences among the interacting configurations are mediated at a given time (not necessarily the equilibrium or steady-state value).

Discussion

The total rate of exchange (Equation 3.32) can be expanded with the rates of advection and diffusion from Equations 3.33 and 3.35:

$$\dot{X}_{ij} = \dot{N}_{ij} \left(\frac{\partial X}{\partial N} \right)_{ij} + \left(\frac{DX}{Dt} \right)_{ij} \quad (3.37)$$

Since the model uses a Eulerian perspective, the total exchange rate \dot{X}_{ij} is actually a partial derivative in the form of $\partial X/\partial t$. Dropping the subscripts and rearranging, the previous equation becomes

$$\frac{DX}{Dt} = \frac{\partial X}{\partial t} - \dot{N} \frac{\partial X}{\partial N} \quad (3.38)$$

This is essentially the definition of a material derivative [160], but in terms of current instead of velocity. Usually, the scalar property in the material derivative is intensive—for example, the diffusion-driving property (γ)—but it is coupled to the exchanged property (X) through the extensive property $\partial X/\partial \gamma$. The usual advective term $\phi \cdot \nabla X$ is a loss. The advective source is $-\phi \cdot \nabla X$ or $\dot{N} \partial X/\partial N$, although the material exchange current itself is purely diffusive (as mentioned previously). The concept here is that particles experience the collisions that lead to diffusive exchange, but they do not experience advection. Advection is an artifact of the Eulerian basis of the model.

There are several types of material exchange processes. The simplest is phase change, which is discussed in the following section. Electrochemical reactions are introduced later (Section 3.9) because they involve geometric dimensions and orientation like the transport equations (also to follow). Chemical reactions are beyond the present scope; they are only intermediate to the electrochemical reactions in a fuel cell.

3.5.1 Phase Change

At equilibrium, a species has the same specific Gibbs energy in each phase. The configurations may equilibrate rapidly [43], yet the model still considers the process to be dynamic. This avoids the nonlinear system of equations that would occur since the Gibbs function (Equation 3.16) is only invertible in certain cases. In terms of Gibbs' phase rule ($n_{\text{DOF}} = 2 + n_{\text{spec}} - n_{\text{phases}}$) [150, 151], we are not subtracting the number of phase equilibria ($n_{\text{phases}} - 1$). Therefore, the number of thermodynamic state variables or degrees of freedom is one plus the number of species ($n_{\text{DOF}} = 1 + n_{\text{spec}}$).^{vii}

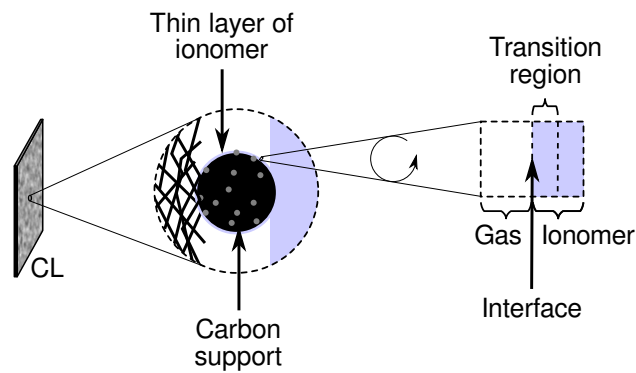
Since the phase change model is dynamic, it is necessary to specify the rate of phase change. One way would be to assume that the rate is proportional to the difference between the vapor pressure and the saturation pressure. Yet this would not avoid nonlinear equations because the saturation pressure is only implicitly known from the thermodynamic properties established in Sections 3.1 and 3.2. We could implement a known correlation for saturation pressure (e.g., [64] or [2]), but this would be redundant and somewhat inconsistent with the existing properties. Another way would be to assume that the rate of phase change is proportional to the differences in specific Gibbs energies between the phases. However, this does not relate well to the classical Hertz-Knudsen equation [161] which establishes the rate in proportion to the difference in adjusted concentrations. Many other approaches could be used [162], but these are more detailed than presently necessary and more complicated than can be efficiently implemented.

The model uses a fairly simple approach that is based on the differences of chemical activities. It is consistent with the generalized equation for diffusive exchange (3.36) with some additional assumptions. It results in linear systems of equations and avoids the need for conditional expressions and dynamic state selection.

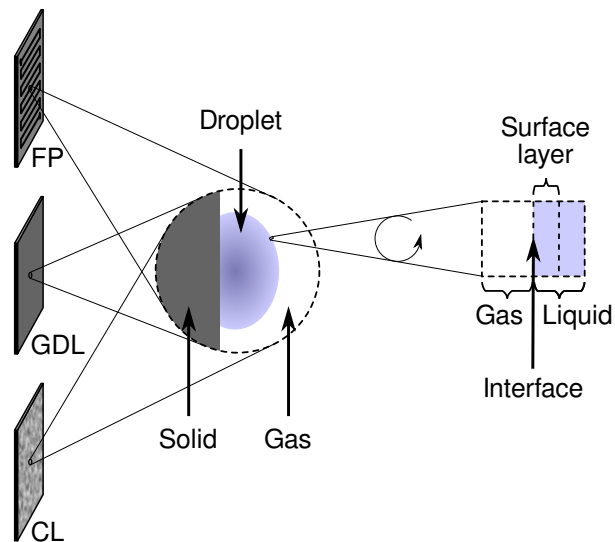
^{vii}In fact, without certain optional assumptions enabled (see Section 4.8) the model often has even more thermodynamic state variables. The number of thermodynamic state variables in the model is the number of species plus the number of compressible species, and there are often several compressible species in a region. In general, the total number of state variables is equal to the number of ways in which energy (not limited to thermal and compressive) may be stored.

3.5.1.1 Context

In a proton exchange membrane fuel cell (PEMFC), water may be absorbed and desorbed between the ionomer and the gas in the catalyst layer, as shown in Figure 3.7a. The catalyst layer extends from the plane where the solid is entirely the gas diffusion layer (GDL) material to the plane where the solid is entirely the ionomer or proton exchange material. Figure 3.7b shows that water may also condense and evaporate between the liquid and gas in the flow plate, the GDL, or the catalyst layer. In any case, there is an interface between the phases. The interface is only a volumeless threshold, but we will assume that there is a transition or surface layer on the condensed side.



(a) Gas to ionomer.



(b) Gas to liquid.

Figure 3.7: Phase change occurs between the gas and (a) the ionomer in the catalyst layers (CLs) and (b) the liquid in the flow plates (FPs), gas diffusion layers (GDLs), and CLs.

3.5.1.2 Equations

The rate of condensation or absorption is given by the diffusive exchange equation (3.36) for material ($X = N$, $\gamma = \rho$, $\partial X / \partial \gamma = V$).^{viii}

$$\frac{8}{3\pi} \tau_c \dot{N}_{Dic} = k_{ic} V_c (\rho_i - \rho_c) \quad (3.39)$$

where the subscript c denotes the condensed or absorbed phase. Since $\rho = N/V$,

$$\frac{8}{3\pi} \tau_c \dot{N}_{Dic} = k_{ic} N_c \left(\frac{\rho_i}{\rho_c} - 1 \right) \quad (3.40)$$

We will assume that the species behaves as an isothermal ideal gas over the transition region or surface layer. Under these conditions, Equation 3.14 evaluates to

$$g_i - g_c = T_c \ln \left(\frac{\rho_i}{\rho_c} \right) \quad (3.41)$$

Therefore, the rate of condensation or absorption can be written as

$$\tau'_{ic} \dot{N}_{Dic} = N_c \left[\exp \left(\frac{g_i - g_c}{T_c} \right) - 1 \right] \quad (3.42)$$

where τ' is the effective collision interval, or the mean time between collisions that yield phase change:

$$\tau'_{ic} = \frac{8}{3\pi k_{ic}} \tau_c \quad (3.43)$$

In practice, τ'_{ic} is an empirical, tunable parameter. Since we have assumed that the entire transition region is within the condensed phase, the gas is in equilibrium with the condition at the interface.

$$g_i = g_g \quad (3.44)$$

where the subscript g denotes the gas phase. Therefore,

$$\tau'_{ic} \dot{N}_{Dic} = N_c \left[\exp \left(\frac{g_g - g_c}{T_c} \right) - 1 \right] \quad (3.45)$$

This can also be written in terms of activity referenced to the condensed phase.

$$\tau'_{ic} \dot{N}_{Dic} = N_c (a_g - a_c) \quad (3.46)$$

^{viii}Phase change is purely diffusive. As mentioned in Footnote v, the advective exchange equation (3.33) is not applicable to material exchange.

which is consistent with the common interpretation of activity as an effective, dimensionless concentration.

This model is appropriate for condensation and evaporation or absorption and desorption. It should be noted that if the condensed phase is entirely absent ($N_c = 0$), there can be no condensation. If phase change is included, the conditions are set so that some amount of material (as slight as it may be) always exists in the condensed or absorbed phase.

If the effective collision interval is zero ($\tau'_{ic} = 0$), then the phases will be in perfect equilibrium. Then, it is no longer a dynamic process, and in general, there will be nonlinear systems of equations. Since phase change is a diffusive process, the equilibration is irreversible. Heat is generated in phase j at the rate of $\dot{N}_{Dij} (g_i - g_j)$ (discussed further in Section 3.10.3). However, this is zero for the gas phase due to Equation 3.44. This excludes the latent heat, which is transferred via thermal advective exchange (Section 3.5.3.1).

3.5.2 Drag and Translational Advection

The translational exchange equations follow from the generalized exchange equations (3.34 and 3.36). The exchanged quantity (X) is the product of the amount of material and velocity (Φ). Both the intensive property $\partial X / \partial N$ and the diffusion-driving property (γ) are velocity (ϕ). The extensive property $\partial X / \partial \gamma$ is the amount of material (N).

3.5.2.1 Advection

Advective translational exchange occurs in a stream of fluid that is undergoing phase change or reaction. It is not significant in the reactions of a PEMFC since they occur at the surface of stationary electrodes. However, the model includes advective translational exchange for completeness; it may be important in other devices.

In terms of translational exchange, Equation 3.34 is the following:

$$\dot{\Phi}_{Aij} = \dot{N}_{ij} \cdot \begin{cases} \phi_j & \text{if } \dot{N}_{ij} < 0 \text{ (source),} \\ \phi_i & \text{if } \dot{N}_{ij} > 0 \text{ (sink)} \end{cases} \quad (3.47)$$

However, the product of the amount of material and velocity (Φ) is not generally conserved. Momentum, $m\Phi$, is. Therefore, we will multiply the previous equation by specific mass so that

it can be written in terms of forces or rates of momentum:

$$\boxed{m\dot{\Phi}_{Aij} = m_j \dot{N}_{ij} \cdot \begin{cases} \phi_j & \text{if } \dot{N}_{ij} < 0 \text{ (source),} \\ \phi_i & \text{if } \dot{N}_{ij} > 0 \text{ (sink)} \end{cases}} \quad (3.48)$$

The variable ϕ_i is the *conversion velocity*, or the velocity at which the products are generated from the reactants during advective exchange. Its value is a consequence of conservation at the interface (Equation 3.29). Since advection and diffusion are independent, the sum of the advective forces ($m\dot{\Phi}_{Aij}$) over all of the interacting configurations is zero. Using Equation 3.48, this is

$$0 = \sum_{j \in \xi_i} m_j \dot{N}_{ij} \cdot \begin{cases} \phi_j & \text{if } \dot{N}_{ij} < 0 \text{ (source),} \\ \phi_i & \text{if } \dot{N}_{ij} > 0 \text{ (sink)} \end{cases} \quad (3.49)$$

where ξ_i is the set of the configurations that interact at transition i . The currents (\dot{N}_{ij}) are related by the stoichiometry of the phase change or reaction.

$$\dot{N}_{ij} = n_{ij} \dot{N}_i \quad (3.50)$$

where \dot{N}_i is the rate of the transition i and n_{ij} is the stoichiometric coefficient of configuration j with respect to transition i . Therefore,

$$0 = \sum_{j \in \xi_i} m_{ij} n_{ij} \cdot \begin{cases} \phi_j & \text{if } n_{ij} \dot{N}_i < 0 \text{ (source),} \\ \phi_i & \text{if } n_{ij} \dot{N}_i > 0 \text{ (sink)} \end{cases} \quad (3.51)$$

Solving for the conversion velocity,

$$\phi_i = \frac{\sum_{j \in \text{source}} |n_{ij}| m_j \phi_j}{\sum_{j \in \text{sink}} |n_{ij}| m_j} \quad (3.52)$$

where the numerator is summed over the sourcing configurations (reactants) and the denominator is summed over the sinking configurations (products). If the process is well-posed, it must conserve mass ($\sum_{j \in \text{source}} |n_{ij}| m_j = \sum_{j \in \text{sink}} |n_{ij}| m_j$), and

$$\phi_i = \frac{\sum_{j \in \text{source}} |n_{ij}| m_j \phi_j}{\sum_{j \in \text{source}} |n_{ij}| m_j} \quad (3.53)$$

Thus, the conversion velocity is the mass-weighted average of the velocities of the configurations consumed by the process.

3.5.2.2 Diffusion

For translational exchange, Equation 3.36 is the following:

$$\frac{8}{3\pi} \tau_j \dot{\Phi}_{Dij} = k_{ij} N_j (\phi_i - \phi_j) \quad (3.54)$$

This can be written as

$$\mu_j m \dot{\Phi}_{Dij} = k_{ij} N_j (\phi_i - \phi_j) \quad (3.55)$$

where μ_j is the mobility:

$$\mu = \frac{8}{3\pi m} \tau \quad (3.56)$$

The variable ϕ_i is the *mediation velocity*. Like the conversion velocity, it can be determined from conservation at the interface. Since advection and diffusion are independent, the sum of the diffusion forces ($m \dot{\Phi}_{Dij}$) over all of the interacting configurations is zero. Using Equation 3.55, this is

$$0 = \sum_{j \in \xi_i} (\phi_i - \phi_j) k_{ij} N_j / \mu_j \quad (3.57)$$

where ξ_i is the set of all the interacting configurations at transition i . Solving for the mediation velocity,

$$\phi_i = \frac{\sum_{j \in \xi_i} \phi_j k_{ij} N_j / \mu_j}{\sum_{j \in \xi_i} k_{ij} N_j / \mu_j} \quad (3.58)$$

This indicates that the mediation velocity is a conductance-weighted average of the velocities of the interacting configurations. The previous equation applies to each set ξ associated with each interaction i . Each set can have a different value of the mediation velocity.

3.5.3 Thermal Conduction and Advection

The translational exchange equations follow from the generalized exchange equations (3.34 and 3.36). The exchanged quantity (X) is heat (Q). The intensive property $\partial X / \partial N$ is the product of specific entropy and temperature ($\partial Q / \partial N = T \partial S / \partial N = Ts$). The diffusion-driving property (γ) is temperature (T). The extensive property $\partial X / \partial \gamma$ is heat capacity (C). The heat capacity is isobaric (C_p), since the pressures of the configurations are assumed to be at equilibrium (see Section 3.3.2).

3.5.3.1 Advection

In terms of thermal exchange, Equation 3.34 is

$$\dot{Q}_{Aij} = \dot{N}_{ij} \cdot \begin{cases} (sT)_j & \text{if } \dot{N}_{ij} < 0 \text{ (source),} \\ (sT)_i & \text{if } \dot{N}_{ij} > 0 \text{ (sink)} \end{cases} \quad (3.59)$$

where j is a configuration that participates in reaction or phase change i . It is important to note that \dot{Q}_A is advective. It is different from \dot{Q}_D , which is the rate of thermal diffusion or conduction. In the energy balance (Equation 3.30), the sT factor combines with the specific Gibbs energy (g) to give the specific enthalpy (h) that is transferred with the process (reaction or phase change). The rate of thermal energy due to $\sum \dot{N}_j (sT)_j$ or $\dot{N} \sum n_j (sT)_j$ over a process (where the subscript i has been dropped) is split stoichiometrically (not by mass) among the products (sinks). The intensive properties of the reactants (sources) are not directly affected due to the conditional factor in Equation 3.59. If the process occurs near equilibrium, then $\sum n_j g_j$ is nearly zero and $\sum n_j (sT)_j$ is nearly $\sum n_j h_j$, the enthalpy of the reaction or phase change. If the process is not at equilibrium, heat is produced at the rate of $\dot{N} \sum n_j g_j$. This is irreversible because the process always occurs towards lower specific Gibbs energy.^{ix}

The property $(sT)_i$ is the thermal conversion property—the product of specific entropy and temperature at which the products are generated from the reactants during advective exchange. Its value is a consequence of conservation at the interface (Equation 3.31). Since advection and diffusion are independent, the sum of the advective rates (\dot{Q}_{Aij}) over all of the interacting configurations is zero. Using Equation 3.59, this is

$$0 = \sum_{j \in \xi_i} \dot{N}_{ij} \cdot \begin{cases} (sT)_j & \text{if } \dot{N}_{ij} < 0 \text{ (source),} \\ (sT)_i & \text{if } \dot{N}_{ij} > 0 \text{ (sink)} \end{cases} \quad (3.60)$$

where ξ_i is the set of the configurations that interact at transition i . The currents (\dot{N}_{ij}) are related by the stoichiometry of the phase change or reaction.

$$\dot{N}_{ij} = n_{ij} \dot{N}_i \quad (3.61)$$

^{ix}This was evident from the equation for the rate of phase change (3.42) and is also the case in electrochemical reactions, inclusive of the electrical work potential (Section 3.9).

where \dot{N}_i is the rate of the transition i and n_{ij} is the stoichiometric coefficient of configuration j with respect to transition i . Therefore,

$$0 = \sum_{j \in \xi_i} n_{ij} \cdot \begin{cases} (sT)_j & \text{if } n_{ij}\dot{N}_i < 0 \text{ (source),} \\ (sT)_i & \text{if } n_{ij}\dot{N}_i > 0 \text{ (sink)} \end{cases} \quad (3.62)$$

Solving for the thermal conversion property,

$$(sT)_i = \frac{\sum_{j \in \text{source}} |n_{ij}| (sT)_j}{\sum_{j \in \text{sink}} |n_{ij}|} \quad (3.63)$$

where the numerator is summed over the sourcing configurations (reactants) and the denominator is summed over the sinking configurations (products). Thus, the thermal conversion property is the stoichiometrically-weighted average of the specific entropy-temperature products (or specific enthalpy-specific Gibbs energy differences) of the configurations consumed by the process.

3.5.3.2 Diffusion

For thermal diffusion (i.e., thermal conduction), Equation 3.36 is

$$\frac{8}{3\pi} \tau_j \dot{Q}_{Dij} = k_{ij} c_{pj} (T_i - T_j) \quad (3.64)$$

This can be written as

$$\boxed{v_j \dot{Q}_{Dij} = k_{ij} N_j (T_i - T_j)} \quad (3.65)$$

where v_j is called thermal *independency* here.^x It is defined by

$$\boxed{v = \frac{8}{3\pi c_p} \tau} \quad (3.66)$$

The variable T_i is the *mediation temperature*. Like the thermal conversion property, it can be determined from conservation at the interface. Since advection and diffusion are independent, the sum of the heat flow rates (\dot{Q}_{Dij}) over all of the interacting configurations is zero. Therefore,

^xThis is the thermal analog of mobility, but there is no established name. It is not called resistivity. Resistivity is resistance times the quotient of area and length, whereas independency is resistance (or *independence*) times the amount of material.

the sum of Equation 3.65 is

$$0 = \sum_{j \in \xi_i} (T_i - T_j) k_{ij} N_j / v_j \quad (3.67)$$

where ξ_i is the set of all the interacting configurations at transition i . Solving for the mediation temperature,

$$T_i = \frac{\sum_{j \in \xi_i} T_j k_{ij} N_j / v_j}{\sum_{j \in \xi_i} k_{ij} N_j / v_j} \quad (3.68)$$

This indicates that the mediation temperature is a conductance-weighted average of the temperatures of the configurations interacting by diffusion. The previous equation applies to each set ξ associated with each interaction i . Each set can have a different value of the mediation temperature.

3.6 Exchange Properties

The base factor in the diffusive exchange properties is the collision interval or the mean time between collisions. It depends on the thermodynamic state and possibly other properties, but it can be estimated from kinetic theory under the following assumptions [148]:

1. The particles are smooth and rigid but elastic spheres with identical radii. This is the “billiard-ball” assumption. It implies that the collisions are instantaneous and conserve kinetic energy.
2. The mean free path, or the average distance a particle travels between collisions, is much larger than the diameter of a particle.
3. The speeds of the particles follow the Maxwell-Boltzmann distribution.

With these assumptions, the mean free path is

$$\lambda = \frac{v}{\sqrt{2} \pi d^2 q} \quad (3.69)$$

where v is the specific volume of the particles (reciprocal of concentration), d is the specific rigid-sphere or Van der Waals diameter, and q is the particle number representing a single particle [148, 163]. The denominator is the product of the intercept area per particle with particles of the same type ($\pi d^2 q$) and a correction due to the Maxwell-Boltzmann distribution

$(\sqrt{2})$.^{xi} The derivation is beyond the present scope (see [148, pp. 31–32] and [163, p. 229]). The collision interval is the mean free path divided by the mean thermal speed ($\sqrt{8T/\pi m}$):

$$\tau = \lambda \sqrt{\frac{\pi m}{8T}} \quad (3.70)$$

The collision interval is called the relaxation time in solid state physics [164]. For a typical species, the collision interval is small. For oxygen as an ideal gas at 25 °C and 21% of atmospheric pressure, the specific volume (ν) is 0.12 m³/mol. The diameter of a particle is approximately 220 pm; therefore the mean free path is approximately 0.9 μm and the collision interval is approximately 2 ns.

The effective collision interval, which is used for phase change, can be determined from the collision interval using Equation 3.43. Mobility and thermal independency can be determined from the collision interval using Equations 3.56 and 3.66. Due to the assumptions implicit in the diffusive exchange equation (3.36), the equations for the effective collision interval, mobility, and thermal independency are only taken to be estimates. However, they are useful if more precise data is not available.

3.7 Transport Equations

Highlights:

- The model describes the transport of every species individually, even in purely advective flow. However, the diffusive exchange of translational momentum (Section 3.9) tends to couple the velocities of the species and thus the rates of advective transport.
- A general transport equation is proposed to handle upstream discretization. It meets the exact solution to a mixed advection/diffusion problem [51].
- The transport equation changes continuously from neutral discretization under pure diffusion to complete upstream discretization in the limiting case of pure advection. This avoids switching events that could slow the simulation if diffusion were not included.

^{xi}It is counterintuitive that the distribution of molecular speeds has an effect on the mean free path, but this has been established in the literature [148, p. 32].

- The model is expressed in resistivity instead of conductivity so that it is well-posed under all representable values.
- The model allows zero or finite dynamic compressibility. The reciprocal, bulk viscosity, is rarely studied [165] and seldom included in fluid simulations [159], let alone fuel cell simulations. The associated effect may be neglected for monoatomic ideal gases and incompressible fluids [157, 159], but following a plausible formation (see Section 3.7.1) the effect is dominant for lightweight particles such as electrons.
- The material transport equation combines the effects of self diffusivity and bulk viscosity to describe material advection and diffusion. That way, the same equations can describe the primarily advective flow down the channels of a PEMFC and the primarily diffusive flow through the layers.

Transport is the transfer of a conserved quantity between adjacent regions. Like exchange (Section 3.5), it is due to advection and diffusion. Advective transport is the transfer of the quantity along with a sustained transfer of material between regions. Diffusive transport is the transfer of the quantity due only to collisions or thermal agitation of the particles, without a sustained material transfer. In diffusion, a particle leaves one region with the specific quantity (or particle-average amount of the quantity) within the configuration and returns with the specific quantity of the other configuration. This brings certain intensive properties—concentration, velocity, and temperature—into equilibrium among the configurations. Since these properties are sufficient to set the thermodynamic state, other properties (e.g., pressure) equilibrate as well.

In transport, unlike exchange, advection and diffusion are not independent. The transport equations are based on the following assumptions:

1. Transport of material, momentum, and energy only occurs between like configurations (e.g., liquid water). Bulk transport is only the net, macroscopic effect of individual species flows. The inter-configurational effects (e.g., between liquid water and water vapor) are described within a region via exchange (Section 3.5).

2. The coordinate system is aligned with the principle axes of transport. For example, if a phase is stratified within a region, the layers must be parallel to one of the planes in the rectilinear grid. This implies that a gradient which induces diffusion along an axis does not induce diffusion along axes orthogonal to it [151].
3. There is no radiative heat transfer.

The total rate of transport of a quantity X into configuration j from boundary i is the sum of the advective and diffusive rates:

$$\dot{X}_i = \dot{X}_{Ai} + \dot{X}_{Di} \quad (3.71)$$

This is the same as for exchange (Equation 3.32) except i stands for a boundary rather than a transition. The subscript j has been removed because the equations of this section all pertain to a single configuration. The advective and diffusive terms will be evaluated separately.

Advection

The rate of advective transport is the product of the current and the change in the transported quantity with respect to the material.

$$\dot{X}_{Ai} = \dot{N}_i \left(\frac{\partial X}{\partial N} \right)_i \quad (3.72)$$

where \dot{N}_i is the total (advective plus diffusive) rate of material transport of a configuration through boundary i into a region. The partial derivative $(\partial X/\partial N)_i$ is an intensive property at the boundary. This equation is general; in fact, it is the same as for advective exchange (Equation 3.33, without subscript j).

Diffusion

Like for exchange (Section 3.5), we can consider the diffusive transport rate to be the material derivative or the transport rate experienced by the particles themselves.^{xii}

$$\dot{X}_{Di} = \left(\frac{DX'}{Dt} \right)_i \quad (3.73)$$

where $(Dt)_i$ is the time for a particle to pass from the boundary to the center of the region. It is the product of (1) the mean collision interval (τ), (2) a logistic function for upstream

^{xii}See the discussion at the end of this section.

discretization ($1/(1 + \exp(\mp Pe/2))$), described below), (3) a factor of three due to the geometry of randomly-moving particles striking a boundary [148, p. 33–35], (4) a factor to account for the effect of other phases on transport length and available cross-section area (k), and (5) the number of collisions required to span the distance from the boundary to the center of the region. That number is the reciprocal of the Knudsen number (Kn) or the length across the region divided the mean free path ($1/Kn = L/\lambda$). The effective difference in the transported quantity $(DX')_i$ is the actual difference $(DX)_i$ divided by the number of collisions required to span the distance between the boundary and the center (again, the reciprocal of the Knudsen number). We will assume that the transported quantity is linear with respect to the driving property (γ); therefore, $(DX)_i = (\gamma_i - \gamma)(\partial X/\partial \gamma)$. The following assumptions have also been implied:

1. The collision events are frequent enough for the average collision interval to be meaningful. This implies that the mean free path, or the average distance traveled between collisions, is much smaller than the length scale of the problem. It is not the case for example in effusion.
2. Between collisions the particles have no influence on one another.
3. The properties carried by a particle depend only on those of the last particle with which it collided.

It follows that

$$3\tau \dot{X}_{Di} = k_i Kn_i^2 \frac{\partial X}{\partial \gamma} (\gamma_i - \gamma) \left(1 + e^{\mp Pe_i/2}\right) \quad (3.74)$$

where the negative of \mp is for the negative side of the region along an axis and the positive is for the positive side.^{xiii} This can be written as

$$R_i \dot{X}_{Di} = (\gamma_i - \gamma) \left(1 + e^{\mp Pe_i/2}\right) \quad (3.75)$$

where the generalized resistance is

$$R_i = \frac{3\tau}{k_i Kn_i^2} \left/ \frac{\partial X}{\partial \gamma} \right. \quad (3.76)$$

^{xiii}This convention is used throughout the chapter. Likewise, the \pm operator indicates that the positive is for the negative side of the region and the negative is for the positive side.

The resistance can be written as

$$R_i = \frac{rL_i}{k_iA_i} \quad (3.77)$$

where the resistivity is

$$r = \frac{3\tau_i V}{\lambda_i^2} \left/ \frac{\partial X}{\partial \gamma} \right. \quad (3.78)$$

using the definition of the Knudsen number ($Kn \equiv \lambda/L$). In terms of resistivity, the diffusive transport equation is:

$$r\dot{X}_{Di} = \frac{k_iA_i}{L_i} (\gamma_i - \gamma) \left(1 + e^{\mp Pe_i/2} \right) \quad (3.79)$$

If two adjacent regions have zero resistivity, their intensive driving properties (γ) will be equal. The transported quantity will flow between the regions without loss in order to meet that requirement and the number of dynamic states will be reduced by one.

The model uses resistance and resistivity instead of conductance and conductivity (the reciprocals) because the equations are numerically well-posed for zero resistivity and resistance but not for zero conductance and conductivity. Values of zero can be directly represented in the modeling language but infinite values cannot (see Chapter 4). Infinite values of resistivity (zero conductivity) can be represented by directly imposing zero flow rate (by a disconnected interface; see Chapter 4).

The logistic factor $1/(1 + \exp(-Pe/2))$ introduces upstream discretization. It changes the effective transport length depending on the direction and the magnitude of advection. The Péclet number, Pe , is the dimensionless rate of advection from the interaction into the configuration. The factor of one half appears in the exponential because, by default, half of the resistance is to either side of the region along the axis of transport. In the Péclet number, the rate of advection is normalized by the rate of diffusion:

$$Pe \equiv \dot{X}_A / \dot{X}_D \quad (3.80)$$

Using the rate of advection from Equation 3.72 and the rate of diffusion across a region (Equation 3.75 as in Equation 3.98) at a differential level along with the definition of resistance (Equation 3.77),

$$Pe = \frac{R\dot{N}}{d\gamma} \frac{\partial X}{\partial N} \quad (3.81)$$

where the subscripts have been dropped. Taking the current to be advective ($\dot{N} = \phi \partial N / \partial x = \phi \partial N / L$),

$$Pe = \frac{R\phi}{L} \frac{\partial X}{\partial \gamma} \quad (3.82)$$

In terms of resistivity,

$$Pe = \frac{r\phi}{kA} \frac{\partial X}{\partial \gamma} \quad (3.83)$$

where $\partial X / \partial \gamma$ is an extensive material property. This implies that the Péclet number is extensive as well. Its magnitude increases as the length of the region along the transport axis increases.^{xiv}

The Péclet number tends to zero at the differential level of discretization or as the velocity becomes zero. Then, the transport equation (3.75) reduces to a typical diffusion relationship.

$$R_i \dot{X}_i = 2(\gamma_i - \gamma) \quad (3.84)$$

where the flow rate is evaluated at one side rather than between the locations of the properties. The factor of two appears because the property difference ($\gamma_i - \gamma$) spans only half the length of the region. We can consider the factor of $1/(1 + e^{\mp Pe/2})$ in the general transport equation (3.75) to be a length scaling factor which is one half in this case. With a length factor of one half, the transport equation (e.g., 3.79) implements the central difference scheme [166].

As the Péclet number becomes large, the length factor becomes one for the upstream side and zero for the downstream side. The property in the region is weakly coupled to the upstream boundary and strongly coupled to the downstream boundary. Equivalently, from the perspective of a boundary, the value of the property at the interface between regions is nearly the value in the upstream region. The downstream region determines the diffusion rate. The limiting cases are listed in Table 3.2, where the subscript n indicates the negative side of a region along an axis and p indicates the positive side. As stated by Patankar [51],

“It is true that the one-way nature of a space coordinate is a one-way process, but diffusion (which is always present) has two-way influences. However, when the flow rate is large, convection overpowers diffusion and thus makes the space coordinate nearly one-way.”

Here, the term “advection” is used instead of “convection” because convection is the serial combination of diffusion and advection, at least in the thermal context.

^{xiv} Although length appears in the denominator of Equation 3.82, it is canceled by the length factor of the resistance which is in the numerator. This leaves $\partial X / \partial \gamma$, which is extensive.

Table 3.2: Limiting cases of the transport equation.

Péclet Number ($Pe_n = Pe_p$)	Negative-side equation	Positive-side equation
$-\infty$	$\gamma_n = \gamma$	$R\dot{X}_p = \gamma_p - \gamma_n$
0	$R\dot{X}_n = 2(\gamma_n - \gamma)$	$R\dot{X}_p = 2(\gamma_p - \gamma)$
∞	$R\dot{X}_n = \gamma_n - \gamma_p$	$\gamma_p = \gamma$

If the resistivity is infinite, the transport equation reduces to the upwind scheme (also known as the upwind-difference scheme, upstream-difference scheme, and donor-cell method) [51]. The exponential switches immediately upon flow reversal. The property at the interface is exactly the property of the upstream region. The downstream region imposes the diffusion rate, which is zero.

As long as the resistivity is finite, the property changes continuously between the limiting cases of purely advective flow in each direction. If we implement the transport equation (3.75) twice—once for each boundary of a region along an axis—and place the restriction that the quantity is not stored in the region ($0 = \dot{X}_n + \dot{X}_p$), then

$$\gamma = \frac{\gamma_p \left(1 + e^{Pe_p/2}\right) + \gamma_n \left(1 + e^{-Pe_n/2}\right)}{2 + e^{Pe_p/2} + e^{-Pe_n/2}} \quad (3.85)$$

If the Péclet numbers of the two boundaries are equal ($Pe = Pe_n = Pe_p$), then

$$\frac{\gamma - \gamma_n}{\gamma_p - \gamma_n} = \frac{1}{1 + e^{-Pe/2}} \quad (3.86)$$

where the right side is the logistic function of $Pe/2$. Figure 3.8 shows the sigmoid curve which it represents. In a similar manner, since there is no storage at an interface (as required by Equations 3.26, 3.29, and 3.31), the property at the interface between two regions is given by

$$\frac{\gamma - \gamma_1}{\gamma_2 - \gamma_1} = \frac{R_1 \left(1 + e^{-Pe/2}\right)}{R_1 \left(1 + e^{-Pe/2}\right) + R_2 \left(1 + e^{Pe/2}\right)} \quad (3.87)$$

where γ (without a subscript) now represents the value of the property at the interface rather than the value in a region. The subscript 1 indicates the first region and 2 indicates the second region. Positive Péclet numbers are directed from the first to the second region. If the regions have identical resistances ($R_1 = R_2$), this reduces to the same logistic function as for a single

region except reflected over the $Pe = 0$ axis:

$$\frac{\gamma - \gamma_1}{\gamma_2 - \gamma_1} = \frac{1}{1 + e^{Pe/2}} \quad (3.88)$$

This means that an interface's property is biased towards the source whereas a region's property is biased towards the exit. Figure 3.9 shows the relation in contrast to Figure 3.8.

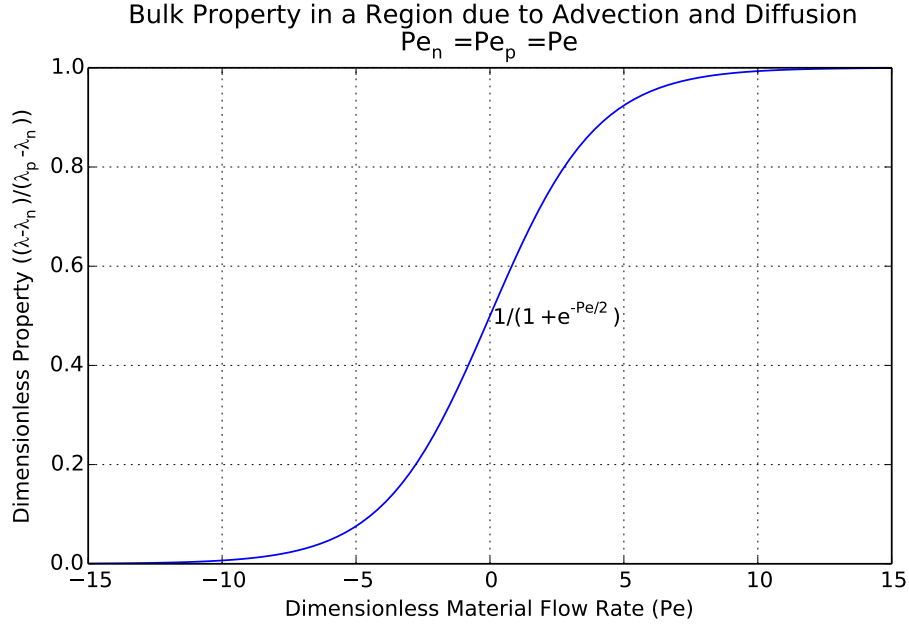


Figure 3.8: Property in the bulk of a region due to advection and diffusion.

Equation 3.87 can be used to provide the profile of the property between the centers of adjacent regions. If we assume that the resistivity and cross-sectional area are uniform, Equation 3.76 implies that

$$\frac{\gamma - \gamma_1}{\gamma_2 - \gamma_1} = \frac{L_1 (1 + e^{-Pe/2})}{L_1 (1 + e^{-Pe/2}) + L_2 (1 + e^{Pe/2})} \quad (3.89)$$

where L_1 is the length across region 1 and L_2 is the length across region 2. If we vary the position of the interface while keeping the center-to-center distance the same (constant $L = (L_1 + L_2)/2$), we can express the value of the property as a function of position.

$$\frac{\gamma - \gamma_1}{\gamma_2 - \gamma_1} = \frac{x^* (1 + e^{-Pe/2})}{x^* (1 + e^{-Pe/2}) + (1 - x^*) (1 + e^{Pe/2})} \quad (3.90)$$

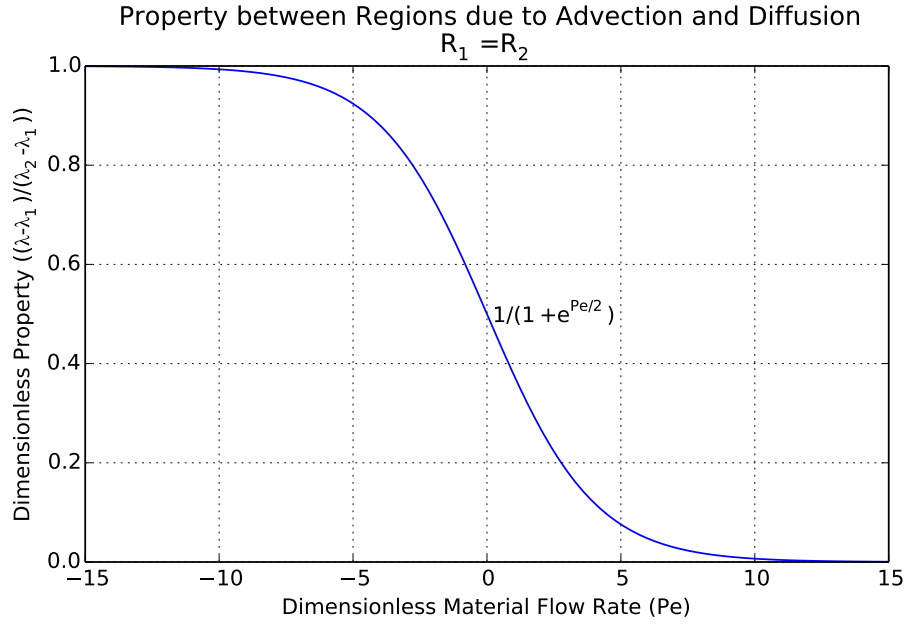


Figure 3.9: Property at the interface between regions due to advection and diffusion.

where $x^* = (x - x_1)/L$ is the dimensionless position between the first and second region as a function of x , the absolute position of the interface. This can be simplified to

$$\frac{\gamma - \gamma_1}{\gamma_2 - \gamma_1} = \frac{1}{1 + \left(\frac{1}{x^*} - 1\right) e^{Pe/2}} \quad (3.91)$$

Figure 3.10 shows the resulting profile for various Péclet numbers (solid lines). The profile is linear under pure diffusion ($Pe = 0$). Otherwise, the profile is biased towards the property of the source. The profile increases or decreases monotonically. Equation 3.91 reduces to Equation 3.88 when $x^* = 1/2$, as shown in the figure.

For comparison, Patankar [51] provides the solution to the following general advection/diffusion equation under the condition of no material storage due to advection ($dI/dx = 0$) and no storage of the transported quantity due to combined advection and diffusion ($d\dot{X}/dx = 0$).

$$\dot{X} = \gamma I - \frac{A}{r} \nabla \gamma \quad (3.92)$$

The equation has been refactored here under the assumption of uniform cross-sectional area.

The solution is

$$\frac{\gamma - \gamma_1}{\gamma_2 - \gamma_1} = \frac{1 - e^{Pe x/L}}{1 - e^{Pe}} \quad (3.93)$$

where L is the center-to-center distance between regions and x is the position. Note that this equation contains a numerical singularity in the case of pure diffusion ($Pe = 0$). It matches Equation 3.91 when $x^* = 1/2$, as shown by Figure 3.10. However, the model and the Patankar's solution are different at other positions. This may be due to one of the following reasons: (1) the model is based on the requirement that the flow rate of the quantity out of one region is the flow rate into the other ($\dot{X}_{1p} + \dot{X}_{2n} = 0$ at the interface plane) whereas Patankar's solution is based on the requirement that there is no storage in a differential space around the interface ($d\dot{X}/dx = 0$) or (2) the assumption of equal Péclet numbers (used in the derivation of Equation 3.91 from Equation 3.87) is unreasonable. The Péclet number is extensive in nature (as mentioned previously), so it may not be appropriate to assume that it remains equal as the adjacent regions are resized.

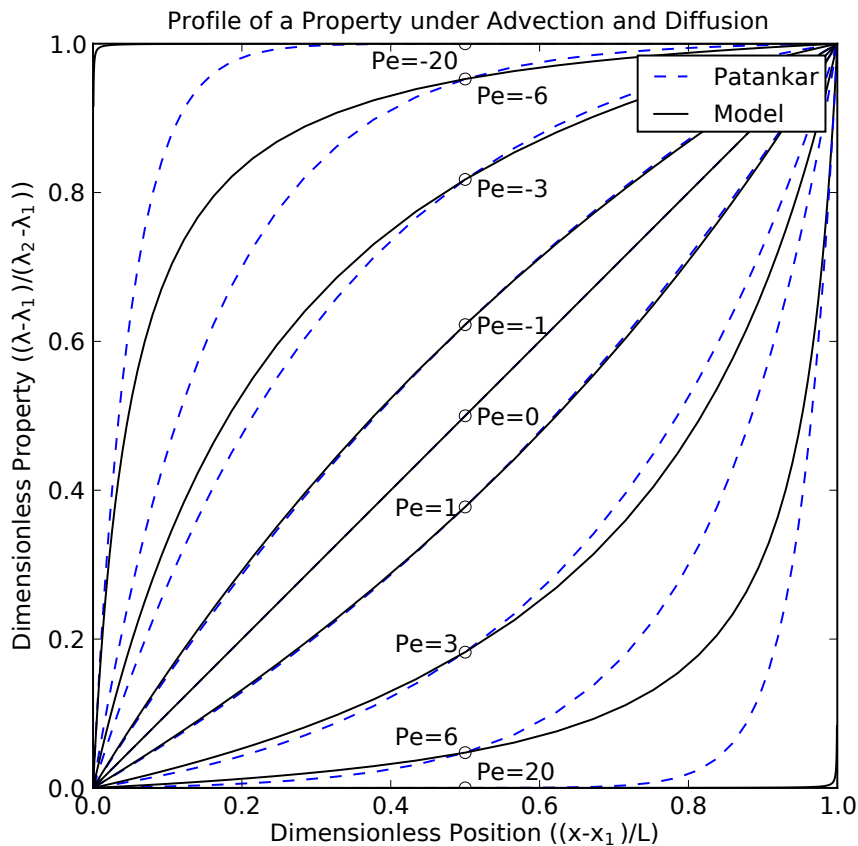


Figure 3.10: Center-to-center profile of a property between regions under advection and diffusion. The model is equivalent to the Patankar's solution [51] at the midplane ($x = L/2$).

The previous evaluations are based on the condition that the flow rates of opposing transport equations are equal and opposite. This is true at an interface between regions because the flow rate into one region is the rate out the other. However, it is not necessarily the case across a region because the quantity may be stored within the region. If we relax the previous assumption and provide the values of the driving property in the bulk of the region and at both boundaries, we can determine the rate of storage in a region:

$$\sum \dot{X} = \frac{(\gamma_n - \gamma) \left(1 + e^{-Pe_n/2}\right) + (\gamma_p - \gamma) \left(1 + e^{Pe_p/2}\right)}{R} \quad (3.94)$$

If there is no advection, the rate of storage is proportional to the first-order approximation of the second derivative of the property over space:

$$\sum \dot{X} = \frac{2}{R} (\gamma_n + \gamma_p - 2\gamma) \quad (3.95)$$

Table 3.3 shows the implications of Equation 3.94. The third textual column and first graphic column indicate the rate of storage induced by positive or negative velocities and positive or negative property gradients under the condition of a linear property profile. The fourth textual column and second graphic column indicate the concavity of the profile under the condition of no storage. The curves are not to scale; Figure 3.10 gives the exact shape. The boundary-to-boundary profile across a region must either match the first or second graphic column (and third or fourth textual column). The center-to-center profile of a property must match the second graphic column—not the first since there is no storage at the interface between regions.

The first row of Table 3.3 indicates that if the property increases in the positive direction and the velocity is in the negative direction, either the conserved quantity is being removed from the region or the profile is concave up, or both. If the gradient or the velocity is reversed, but not both, the quantity is stored instead or the concavity changes sign. If both the gradient and the velocity are reversed, the storage regime and the concavity remain the same. If the material flow is from higher to lower values of the property, the quantity is removed from the region; otherwise it is stored. The concavity is always such that the gradient is lower on the side receiving the advection.

Table 3.3: Scenarios of 1D advection with diffusion.

Difference ($\Delta\gamma$)	\wedge	Velocity (ϕ)	\Leftrightarrow	Intake ($\sum \dot{X}$)	\vee	Inflection ($\Delta^2\gamma$)	Graphically: $\gamma_n \cdots \gamma \cdots \gamma_p$ or
< 0		< 0		> 0		< 0	
< 0		> 0		< 0		> 0	
> 0		< 0		< 0		> 0	
> 0		> 0		> 0		< 0	

The difference between the properties of adjacent regions is related to the flow rate between them by

$$\gamma_2 - \gamma_1 = \dot{X} \left(\frac{R_1}{1 + e^{Pe/2}} + \frac{R_2}{1 + e^{-Pe/2}} \right) \quad (3.96)$$

This follows from Equation 3.75 (implemented for each region) and conservation of the transported quantity at the interface. If the resistances are equal ($R_1 = R_2 = R$), then

$$R\dot{X} = \gamma_1 - \gamma_2 \quad (3.97)$$

which is typical diffusion. This is applicable even if there is bulk material flow. It confirms that the transport equation is a diffusion equation—only with upstream discretization so that advection can be properly determined. Since the local gradient is affected by advection, the rate of diffusion is generally not proportional to the local gradient at the interface (given by Equation 3.91) but rather the average gradient between the centers of the regions. Likewise, if there is no storage within a region along an axis and the Péclet numbers are equal at the boundaries, then

$$R\dot{X} = \gamma_n - \gamma_p \quad (3.98)$$

where $\dot{X} = \dot{X}_n = -\dot{X}_p$.

Discussion

The rate of advection is the product of the material flow rate and the amount of the quantity per unit of material (i.e., specific quantity). In the case of material flow, this specific quantity

is unity, since material is the quantity [148]. In the case of the translational advection, the specific quantity is velocity, the gradient of which also drives property. For thermal advection, the specific quantity is temperature times specific entropy. Temperature is the driving property for thermal diffusion, but specific entropy can be calculated from the temperature and concentration at the boundary.

Figure 3.11 shows the combined effects of advection and diffusion if the specific quantity is the same as the driving property for diffusion (e.g., translational advection). The advection and diffusion are evaluated at the interface of two regions with the same resistances. The property in the second region (γ_1) is arbitrarily five times the property in the first region (γ_2). The rate of diffusion is constant (in dimensionless units), as predicted by Equation 3.97. As advection is initially increased, the property at the interface becomes nearly the mean of the properties of the regions. The actual rate of advection is quite different from the ideal rate given by the upstream scheme. A correction must be applied because the property at the interface is not the property of the upstream region. Here, that correction is called *irreversible advection* because this part of advection has been affected by the irreversible process of diffusion. As the magnitude of the Péclet number becomes large, the irreversible advection becomes negligible because the interface takes on the property of the upstream region.

So far, we have evaluated the transport equations along one axis. It is possible to implement Equation 3.75 for all three dimensions simultaneously as long as the coordinate system is aligned with the principle axes of transport (Assumption 2 in the beginning of Section 3.7). When the coordinate system is not aligned with the principle axes of transport, the model is subject to false diffusion like most methods of upstream discretization [51].

Although the transport equation (3.75) contains an exponential, it is not equivalent to the exponential scheme. The exponential scheme is derived by (1) solving the advection/diffusion equation analytically for the profile of the driving property under the assumption of no storage and (2) reintroducing the solution into the advection/diffusion equation without that original assumption [51, 166]. It results in a solution that is numerically singular unless there is advection. The main argument against the exponential scheme—that it is computationally expensive—also applies to the transport equation used here. However, the transport equation

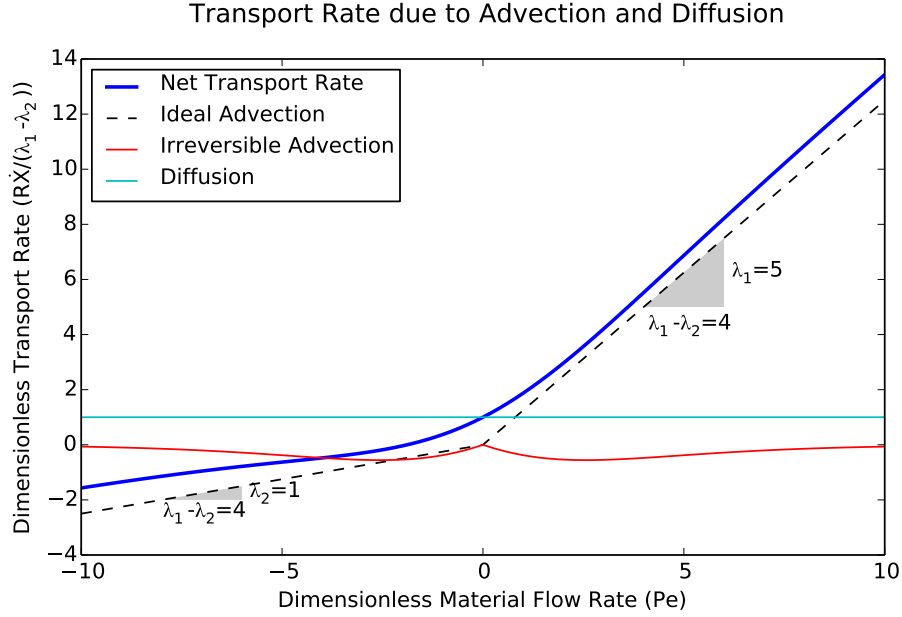


Figure 3.11: Transport rate of a conserved quantity under mixed advection and diffusion.

is used within a framework that (1) does not require a large number of regions for convergence and (2) eliminates all nonlinear systems of equations. The argument may also be out of context now because it dates back to at least 1980 [51], when computational resources were relatively limited.

As demonstrated in Section 3.5, the traditional material derivative is the rate of diffusion. This holds for transport as well. The total rate of transport (Equation 3.71) can be expanded with the rates of advection and diffusion from Equations 3.72 and 3.74:

$$\dot{X}_i = \dot{N}_i \left(\frac{\partial X}{\partial N} \right)_i + \left(\frac{DX}{Dt} \right)_i \quad (3.99)$$

We will take this to be the positive side of the region so that the derivatives are in the positive direction and assume that the current is advective ($\dot{N} = -\phi A \rho = -\phi \partial N / \partial x$). Then, dropping the subscripts,

$$\dot{X} = -\phi \frac{\partial X}{\partial x} + \frac{DX}{Dt} \quad (3.100)$$

Since the model uses a Eulerian perspective, the total transport rate \dot{X} is actually a partial derivative in the form of $\partial X / \partial t$. After rearranging,

$$\frac{DX}{Dt} = \frac{\partial X}{\partial t} + \phi \frac{\partial X}{\partial x} \quad (3.101)$$

Usually the scalar property in the material derivative is intensive (e.g., γ , the driving property for diffusion). Dividing by $\partial X/\partial \gamma$,

$$\frac{D\gamma}{Dt} = \frac{\partial \gamma}{\partial t} + \phi \frac{\partial \gamma}{\partial x} \quad (3.102)$$

and generalizing to three dimensions,

$$\frac{D\gamma}{Dt} = \frac{\partial \gamma}{\partial t} + \boldsymbol{\phi} \cdot \nabla \gamma \quad (3.103)$$

This is the definition of the material derivative [160].

3.7.1 Material Transport

As stated by James Clerk Maxwell, “Mass transfer is due partly to the motion of translation and partly to that of agitation” [167]. Here, the “translation” component of mass transfer is called material advection. It is also called migration in the context of chemical engineering and drift in the context of solid state physics (Section A.4). The “agitation” component is diffusion. In accordance with Maxwell’s statement, the total current into the region through boundary i is the sum of the advective (“translation”) and diffusive (“agitation”) currents:

$$\dot{N}_i = \dot{N}_{Ai} + \dot{N}_{Di} \quad (3.104)$$

This is simply Equation 3.71 where the transported quantity (X) is material (N).

3.7.1.1 Advection

The general advection equation (3.72) reduces to an identity when the transported quantity (X) is the amount of material (N). Instead, the amount of material in that equation is first replaced by the position into the region along the axis of transport. Then, the partial derivative $\partial X/\partial N$ becomes $\partial N/\partial x$ or ρA . The current \dot{N} becomes \dot{x} or velocity directed into the region ($\pm \phi$). Therefore,

$$\dot{N}_{Ai} = \pm \phi_{\perp i} \rho_i A_i \quad (3.105)$$

where the positive of \pm is for the negative boundaries and the negative is for positive boundaries. The subscript \perp indicates the component of velocity normal to the boundary. The value of the velocity will be established by Equation 3.123, but for now, it is considered known.

3.7.1.2 Diffusion

The rate of material diffusion follows from Equations 3.79 and 3.82. Material (N) is the transported quantity (X) and concentration (ρ) is the driving property (γ).

$$\zeta \dot{N}_{Di} = \frac{k_i A_i}{L_i} (\rho_i - \rho) \left(1 + e^{\mp \zeta V \phi_{\perp i} / 2k_i A_i} \right) \quad (3.106)$$

where \dot{N}_i is the diffusive current into the region through boundary i . The volume (V) is the volume of the phase. However, the area (A) is the cross-sectional area of the region along the axis of transport and L is the length of the same. The variable ρ_i is the concentration or volumic amount of material at the boundary. This concentration is not molar concentration or number concentration; it can be determined independently for each species. The concentration in the region may be considered known, since it is a state of the material balance (Section 3.4.1). If a species is isochoric in a phase (e.g., liquid H_2O), it will not diffuse.

The material resistivity, ζ , can be estimated from the generalized definition of resistivity (Equation 3.78):

$$\zeta = \frac{3\tau}{\lambda^2} \quad (3.107)$$

The material resistivity is the reciprocal of self diffusivity, which is the ability of trace particles of a species to diffuse through other particles of the same species [148]. This is the essence of Equation 3.106; it describes the rate of diffusion through an advected stream of particles of the same type.

The material transport equation is closely related to Fick's law. If we assume that the bulk velocity (and advective current) is zero along the axis of transport and the area factor (k_i) is unity, the rate of material diffusion (Equation 3.106) reduces to

$$\zeta \dot{N}_i = 2 \frac{A_i}{L_i} (\rho_i - \rho) \quad (3.108)$$

If the concentration gradient is uniform, there is no material storage due to diffusion ($\dot{N}_n = -\dot{N}_p = AJ$). In that case, we can combine the negative- and positive-side equations as:

$$\zeta J = \frac{\rho_n - \rho_p}{L} \quad (3.109)$$

Taking the limit as length goes to zero and generalizing to three dimensions,

$$\zeta \mathbf{J} = -\nabla \rho \quad (3.110)$$

which is Fick's first law [155, 167–171].

Fick's law also appears in other forms in the literature. In theory, any intensive thermodynamic property could be used as long as it is sufficient to fix the thermodynamic state of the species given its temperature. The choice affects the transport rate and thus the resistivity must be set properly, but it will not affect the equilibrium. All of the intensive thermodynamic properties are uniform between two regions in thermodynamic equilibrium (aside from outside forces), and the equilibrium point is determined by the conservation equations (Sections 3.4 and 3.10).

Since the species has constant specific mass, we can write Fick's law in terms of density [151, 167]:

$$m\zeta J = -\nabla (m\rho) \quad (3.111)$$

If the species exists in small amounts within an otherwise uniform mixture, the extensive volume (in $\rho = N/V$) can be replaced by the total amount or mass of the phase. For mole fractions, this is [166, 167, 171, 172]

$$\frac{V\zeta}{N_{\text{tot}}} J \approx -\nabla (N/N_{\text{tot}}) \quad (3.112)$$

where N is the number of particles of the transported species and N_{tot} is the total number of particles in the phase. For mass fractions, we can write this as [166, 167, 171, 172]

$$\frac{mV\zeta}{M_{\text{tot}}} J \approx -\nabla (M/M_{\text{tot}}) \quad (3.113)$$

We can also write Fick's law in terms of partial pressure [167]:

$$\left(\frac{\partial p}{\partial \rho} \right)_T \zeta J = -\nabla p \quad (3.114)$$

For ideal gases, this is

$$T\zeta J = -\nabla p \quad (3.115)$$

Since the material resistivity is roughly the base resistivity factor divided by specific volume (see Equation 3.107), we can write this as (again, assuming ideal gas)

$$\alpha J = -\nabla \ln p \quad (3.116)$$

For an isothermal ideal gas, Equation 3.14 implies that

$$T\alpha J = -\nabla g \quad (3.117)$$

Since α is only dependent on temperature and fixed properties (specific mass and particle diameter), chemical potential (here, g) may be preferred as the driving property for material diffusion. The magnitude of the diffusion rate in the previous forms of Fick's law is coupled to the concentration, which is affected over time by the diffusion itself (if storage is allowed). We can also express Fick's law in terms of chemical potential for species that are not ideal gases [173]:

$$\zeta \left(\frac{\partial g}{\partial \rho} \right) J = -\nabla g \quad (3.118)$$

If temperature is uniform, this is

$$\frac{\zeta}{\rho} \left(\frac{\partial p}{\partial \rho} \right)_T J = -\nabla g \quad (3.119)$$

If the species is an ideal gas, this again reduces to Equation 3.117. The model uses concentration as the driving property for diffusion because the boundary pressure is needed for the momentum balance (Section 3.10.2) and concentration is available as a dynamic property anyway. Pressure is an explicit function of concentration as long as the species is compressible (Equation 3.6), but pressure cannot generally be expressed as a function of specific Gibbs energy (except in special cases).

3.7.2 Normal Translational Advection and Nonequilibrium Compression

In the previous section, we treated the normal component of velocity at a boundary ($\phi_{\perp i}$) as if it were known. In this section, it is related to the velocity within the region by the effect of bulk or volume viscosity. This entails a set of advection/diffusion equations for the normal component of translational momentum.

The normal force on boundary i is due to the sum of the advective and diffusive forces:

$$m\dot{\Phi}_{\perp i} = m\dot{\Phi}_{A\perp i} + m\dot{\Phi}_{D\perp i} \quad (3.120)$$

This follows the form of Equation 3.71 where the transported quantity (X) is the normal component of translational momentum ($m\Phi$). The advective component is often called dynamic

pressure (or force in this case) and the diffusive component may be called nonequilibrium pressure [159]. Thermodynamic pressure is excluded here, but it is included in the translational momentum balance (Equation 3.28).

3.7.2.1 Advection

The advective normal force follows from Equation 3.72.

$$\dot{m}\Phi_{A\perp i} = \dot{N}_i m \phi_{\perp i} \quad (3.121)$$

where $\phi_{\perp i}$ is the normal velocity at boundary i . Since this equation applies to each configuration separately, the specific mass at the boundary (m_i) is the same as the specific mass in the region (m) and the subscript is not necessary. The current (\dot{N}_i) includes advective and diffusive parts according to Equation 3.104.

Using Equations 3.104 and 3.105, the advective normal force can be expanded as

$$\dot{m}\Phi_{A\perp i} = m\dot{N}_{Di}\phi_{\perp i} \pm m\rho_i A_i \phi_{\perp i}^2 \quad (3.122)$$

where x is the position along the axis of transport. The second term on the right side is closely related to the dynamic pressure ($m\rho\phi^2/2$) usually expressed in partial differential equations (PDEs). However, the essence of dynamic pressure—the advection of translational momentum—is implemented directly instead of using a discrete approximation to the traditional PDEs. Assuming that the specific mass and concentration are constant, the derivative of the traditional dynamic pressure is $m\rho\phi\partial\phi$. The force that results over a distance dx is $M\phi\partial\phi/\partial x$. The first-order discrete approximation is $M\phi\Delta\phi/L$; its implementation would yield a boundary force of $\pm m\rho A\phi_{\perp}\phi_{\perp i}$ in the nomenclature of Equation 3.122. This is only an approximation to the force due to material advection. It does not guarantee conservation of momentum at the boundary, since adjacent regions may have different normal velocities (ϕ_{\perp}). This is troubling because the lack of conservation can lead to artificial instabilities. The actual force—the one used in the model—involves the product of density, advective current, and velocity precisely at the interface (second term on the right side of Equation 3.122).

3.7.2.2 Diffusion

The diffusive normal force follows from Equation 3.79 with the component of velocity normal to the boundary (ϕ_{\perp}) as the diffusion-driving property (γ):

$$\beta m \dot{\Phi}_{D_{\perp i}} = \frac{2k_i A_i}{L_i} \left[\phi_{\perp i} - \phi_{\perp} \frac{V}{V_{\text{tot}}} \right] \quad (3.123)$$

This is the force due to nonequilibrium compression. The form of the equation departs from the general equation (3.79) in two ways, which will be explained.

The first departure from Equation 3.79 is that the porosity, V/V_{tot} , has been applied to the velocity in the region to account for the presence of other phases. If multiple phases are present in the region, only part of the boundary is open to material transport in any one of the phases. Yet the material advection equation (3.105) assumes that the velocity is uniform over the boundary. The porosity is an attempt to account for this inaccuracy. The fraction of open area at the boundary (e.g., areal porosity) is assumed to be the fraction of open volume in the region (e.g., volumetric porosity). The porosity cannot be introduced directly in the material advection equation because neighboring regions may have different volume fractions. If it were, material conservation would be violated at the boundary. By placing the porosity in Equation 3.123, the neighboring regions must agree on the fraction of open area at their common boundary.

The second departure is that no upstream discretization has been applied in Equation 3.123. The velocity at the boundary is assumed to be zero for the sake of calculating the Péclet number at the boundary (Equation 3.82). A nonzero Péclet number would imply that the velocity at the boundary anticipates the propagation of the velocity property itself from the center of the region to the boundary. This seems to be a violation of physics.

The variable β represents the *dynamic compressibility* which describes the extent to which a non-equilibrium normal force causes or requires transient compression. It can be estimated from the generalized definition of resistivity (Equation 3.78):

$$\beta = \frac{3\tau}{\lambda^2 \rho m} \quad (3.124)$$

Dynamic compressibility is the reciprocal of bulk viscosity as a dynamic (rather than kinematic) viscosity. Although bulk viscosity differs from shear viscosity [165, 169, 174, 175], the same initial estimate is used. Dynamic compressibility is different from compressibility, which is $-(\partial v/\partial p)/v$.

3.7.3 Transverse Translational Advection and Friction

The force on boundary i transverse to the surface is due to the sum of the advective and diffusive forces:

$$m\dot{\Phi}_{\parallel i} = m\dot{\Phi}_{A\parallel i} + m\dot{\Phi}_{D\parallel i} \quad (3.125)$$

This follows the form of Equation 3.71 where the transported quantity (X) is a component of translational momentum transverse to the boundary ($m\dot{\Phi}_{\parallel}$). This equation is implemented twice—once for each transverse direction.

3.7.3.1 Advection

Translational momentum is advected according to the generalized advective transport equation (3.72). In terms of the present variables, this is

$$m\dot{\Phi}_{A\parallel i} = \dot{N}_i m \phi_{\parallel i} \quad (3.126)$$

where $\phi_{\parallel i}$ is a component of velocity transverse to boundary i . Since this equation applies to each configuration separately, the specific mass at the boundary (m_i) is the same as the specific mass in the region (m) and the subscript is not necessary. The current (\dot{N}_i) includes advective and diffusive parts according to Equation 3.104.

3.7.3.2 Diffusion

The friction or shear force along a boundary follows the generalized diffusive transport equation (3.79) with an adjustment factor. The driving property is a transverse component of velocity (ϕ_{\parallel}) aligned with the shear force.

$$\boxed{\eta m\dot{\Phi}_{D\parallel i}' = \frac{Nu_{\Phi} k_i A_i}{L_i} (\phi_{\parallel i} - \phi_{\parallel}) \left(1 + e^{\mp \eta M \phi_{\perp i} / 2k_i A_i}\right)} \quad (3.127)$$

where $m\dot{\Phi}_{D\parallel i}'$ is the shear force. The reason for the prime superscript will be discussed below. The boundary velocity ($\phi_{\parallel i}$) is an effective velocity. Its usage does not imply that the velocity is uniform over the boundary (which would generally lead to discontinuities in the velocity at the edges of the boundary). Unlike the normal diffusive force (Equation 3.123), the shear force uses upstream discretization. The associated Péclet number is based on the normal component of velocity at the boundary ($\phi_{\perp i}$).

The fluidity, η , can be estimated from the generalized definition of resistivity (Equation 3.78):

$$\eta = \frac{3\tau}{\lambda^2 \rho m} \quad (3.128)$$

Fluidity is the reciprocal of shear viscosity as a dynamic (rather than kinematic) viscosity. If two adjacent regions have zero fluidity (η), the parallel components of their velocities are equal and the number of states is reduced by two. Translational momentum will flow between the regions without loss.

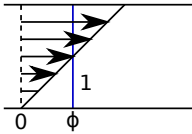
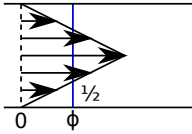
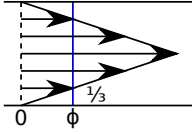
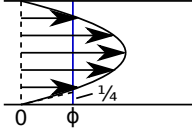
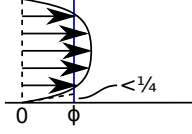
The additional variable Nu_{Φ} is the shear shape factor or the *translational Nusselt number*—the analog of the traditional or thermal Nusselt number for the diffusion of translational momentum. It allows the shear force equation (3.127) to be used at relatively coarse levels of discretization. It is defined as the ratio of the average inward velocity gradient along the perimeter divided by the difference between the boundary velocity and the bulk velocity over the distance from the boundary to the center of the region:

$$Nu_{\Phi} \equiv \left(\frac{\partial \phi_{\parallel}}{\partial x} \right)_i \bigg/ 2 \frac{\phi_{\parallel i} - \phi_{\parallel}}{L_x} \quad (3.129)$$

Table 3.4 summarizes the shape factor or the translational Nusselt number given two assumptions: (1) the material concentration is uniform and (2) the boundary velocities are uniform around the perimeter. The first assumption implies that the bulk, mass-average velocity is equal to the volume-average velocity. The shape factor is one in a two-dimensional (2D) case (where no force is applied the third dimension, e.g., its length is infinite) with a linear velocity profile. The velocity changes from the boundary velocity to the bulk velocity (solid blue vertical line) in half of the distance between the boundaries. The shape factor is two in a two-dimensional (2D) case with a piecewise linear velocity profile. Then, the velocity changes from the boundary

velocity to the bulk velocity in one fourth of the distance between the boundaries. The shape factor is three in a three-dimensional (3D) case with a piecewise linear velocity profile because the volume of a pyramid is one third the product of its base area and height. The shape factor is four if the flow is fully developed and laminar (Hagen-Poiseuille flow, discussed below). If the flow is turbulent, the shape factor is greater than four; the velocity gradient at the wall is greater than for laminar flow.

Table 3.4: Values of the translational Nusselt number.

Shape factor (Nu_ϕ)	Profile	Number of dimensions	Description
1		2	Couette flow (linear)
2		2	Piecewise linear (triangle)
3		3	Piecewise linear (pyramid)
4		3	Laminar fully developed (parabola)
> 4		3	Turbulent (higher order)

The model does not directly use the shear force equation (3.127) because the forces apply torques and the model is based on the assumption that rotational momentum is not stored (see Section 3.4.2). This is the reason for the prime superscript in the shear force equation (3.127). The conservation of rotational momentum (Equation 3.27a) gives one of the four equations required for the four boundaries around the perimeter of the region along the x axis. The first of the remaining three equations requires that the total force in the y direction is the force

determined by the y-direction shear force equations:

$$\boxed{\dot{m}\Phi_{\text{ynz}} + \dot{m}\Phi_{\text{ypz}} = \dot{m}\Phi'_{\text{ynz}} + \dot{m}\Phi'_{\text{ypz}}} \quad (3.130)$$

Similarly in the z direction:

$$\boxed{\dot{m}\Phi_{\text{zny}} + \dot{m}\Phi_{\text{zpy}} = \dot{m}\Phi'_{\text{zny}} + \dot{m}\Phi'_{\text{zpy}}} \quad (3.131)$$

The final equation requires that the torque applied to the y boundaries opposes the torque applied to the z boundaries to the same extent calculated by the shear force equations:

$$\boxed{(\dot{m}\Phi_{\text{zny}} - \dot{m}\Phi_{\text{zpy}}) L_z + (\dot{m}\Phi_{\text{ynz}} - \dot{m}\Phi_{\text{ypz}}) L_y = (\dot{m}\Phi'_{\text{zny}} - \dot{m}\Phi'_{\text{zpy}}) L_z + (\dot{m}\Phi'_{\text{ynz}} - \dot{m}\Phi'_{\text{ypz}}) L_y} \quad (3.132)$$

Similar equations apply to the perimeters around the y and z axes. The translational advection equation (3.126) is applied directly; we assume that it interacts with the center of the region and produces no torque.

In three dimensions, this method involves a system of twelve equations which can be solved for the twelve shear forces (two for each of six boundaries). The shear forces in the x direction are:

$$4\dot{m}\Phi_{\text{ynx}} = 3\dot{m}\Phi'_{\text{ynx}} + \dot{m}\Phi'_{\text{ypx}} + \frac{L_x}{L_y} (\dot{m}\Phi'_{\text{xny}} - \dot{m}\Phi'_{\text{xpy}}) \quad (3.133a)$$

$$4\dot{m}\Phi_{\text{ypx}} = 3\dot{m}\Phi'_{\text{ypx}} + \dot{m}\Phi'_{\text{ynx}} + \frac{L_x}{L_y} (\dot{m}\Phi'_{\text{xpy}} - \dot{m}\Phi'_{\text{xny}}) \quad (3.133b)$$

$$4\dot{m}\Phi_{\text{znx}} = 3\dot{m}\Phi'_{\text{znx}} + \dot{m}\Phi'_{\text{zpx}} + \frac{L_x}{L_z} (\dot{m}\Phi'_{\text{xnz}} - \dot{m}\Phi'_{\text{xpz}}) \quad (3.133c)$$

$$4\dot{m}\Phi_{\text{zpx}} = 3\dot{m}\Phi'_{\text{zpx}} + \dot{m}\Phi'_{\text{znx}} + \frac{L_x}{L_z} (\dot{m}\Phi'_{\text{xpz}} - \dot{m}\Phi'_{\text{xnz}}) \quad (3.133d)$$

Thus, the force applied to a boundary is three fourths of the force calculated directly from Equation 3.127 plus one fourth of the calculated force on the opposing boundary minus the calculated forces on the perpendicular faces which are scaled to oppose the torques implied by Equation 3.127. This is shown graphically in Figure 3.12 for a cubic region ($L_x = L_y = L_z$). The sum of the applied forces is equal to the forces calculated directly from Equation 3.127:

$$\dot{m}\Phi_{\text{ynx}} + \dot{m}\Phi_{\text{ypx}} + \dot{m}\Phi_{\text{znx}} + \dot{m}\Phi_{\text{zpx}} = \dot{m}\Phi'_{\text{ynx}} + \dot{m}\Phi'_{\text{ypx}} + \dot{m}\Phi'_{\text{znx}} + \dot{m}\Phi'_{\text{zpx}} \quad (3.134)$$

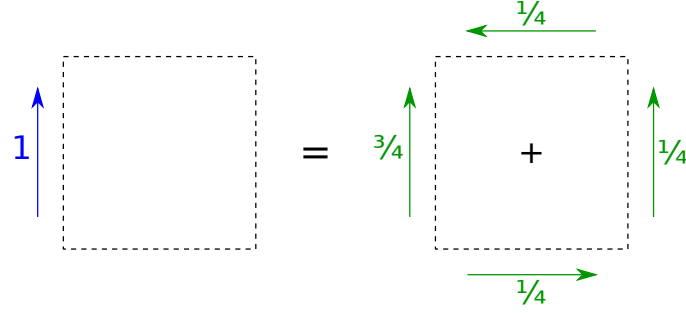


Figure 3.12: Weighting scheme to achieve zero torque. The left side is the force applied to a boundary and the right side contains the forces calculated from Equation 3.127.

Similar equations apply around the y and z axes.

This mapping of shear velocities to shear forces is different from Stokes' viscous deformation law [166]. Both methods impose zero torque (i.e., conservation of rotational momentum without storage), but a discrete implementation of Stokes' viscous deformation law would do so at every boundary. This is not necessary; the boundaries have zero thickness and thus zero moment. However, there is a moment from the boundaries to the center of the control volumes. Thus, the model imposes zero torque on each control volume as a whole.

If the normal bulk velocity is zero, the shear force equation reduces to

$$\dot{m}\Phi_{D\parallel i} = \frac{2Nu_{\phi_i}A_i}{\eta L_i} (\phi_{\parallel i} - \phi_{\parallel}) \quad (3.135)$$

If the velocity gradient is uniform ($Nu_{\phi} = 1$), translational momentum is not stored due to diffusion ($\dot{m}\Phi_{D\parallel n} = -\dot{m}\Phi_{D\parallel p} = AJ$). In that case, we can combine the negative- and positive-side equations as:

$$J = \frac{\phi_{\parallel n} - \phi_{\parallel p}}{\eta L} \quad (3.136)$$

This maps directly to the actual shear stress if there is no shear force from the other two boundaries along the perimeter. It is the first-order approximation to Newton's law of viscous shear, which describes Couette flow [176].

The shear force equation results in the expected pressure loss for fully-developed laminar pipe flow (i.e., Hagen-Poiseuille flow). Suppose that material is flowing in the x direction with velocity ϕ through a region with dimensions L_x , L_y , and L_z . The bulk velocities in the y and z directions are zero and the x-direction velocity is zero at the y and z boundaries (i.e., no slip).

According to Equation 3.127, the total shear force on the y and z boundaries is

$$\dot{m}\Phi_{\text{tot}} = -\frac{4Nu_{\Phi}L_x\phi}{\eta} \left(\frac{L_z}{L_y} + \frac{L_y}{L_z} \right) \quad (3.137)$$

If the flow is fully developed, steady, and laminar, $Nu_{\Phi} = 4$. We may write the equation in terms of the hydraulic diameter ($D \equiv 4A/P$) [172], which is $2L_yL_z/(L_y + L_z)$ in this case. If there are no other forces the shear force must be balanced by the normal force ($L_yL_z\Delta p = \dot{m}\Phi_{\text{tot}}$; see the conservation of translational momentum in Section 3.4.3). With these assumptions and substitutions,

$$\Delta p = \frac{64L_x\phi}{\eta D^2} \left(\frac{2}{2 + \frac{L_y}{L_z} + \frac{L_z}{L_y}} - 1 \right) \quad (3.138)$$

If we assume that the cross section is square ($L_y = L_z$), this reduces to

$$\Delta p = -\frac{32L_x\phi}{\eta D^2} \quad (3.139)$$

which is Poiseuille's law. It can be derived by integrating a differential representation of the shear force equation (3.127) with a translational Nusselt number of one under the assumption of uniform pressure and a no-slip condition around a circular pipe [177]. This implies that the model should give the same result without the shear shape factor (i.e., $Nu_{\Phi} = 1$) under a sufficiently fine level of discretization. As mentioned previously, the shear shape factor is introduced to allow much coarser levels of discretization, and here it is set to four.

The model cannot directly describe turbulence because it assumes that the rotational momentum is zero. This implies that eddies are not generated or are dissipated immediately upon generation. A full description would require equations for the storage, exchange, and transport of rotational momentum. Instead, correlations for turbulent flow are cast into the shear shape factor, which may vary with time and depend on the conditions. This approach is consistent with the statement by Patankar [51]: "From a computational viewpoint, a turbulent flow within this framework is equivalent to a laminar flow with a rather complicated prescription of viscosity." The only difference in this respect is that the present model has an adjustment factor (Nu_{Φ}) which is separate from the fluidity (or reciprocal of viscosity).

If rotational momentum were included, it is plausible that at a sufficiently fine level of discretization the model could describe turbulence. In reality, shear force generates eddies that

drive material towards alternating boundaries around the perimeter—normal to the direction of primary flow. This effect would decrease the advection-adjusted length between the bulk velocity and a boundary (see Equation 3.127) and increase the shear force for any velocity difference. The criteria for the effect is that a sufficient amount of translational momentum is diverted from the direction of primary flow towards a boundary (e.g., by surface roughness). It is a positive feedback mechanism; as more translational momentum is diverted towards the boundary due to shear force, more shear force is produced, until the translational momentum in the primary direction is sufficiently depleted. Suppose we let ω be the diversion angle at the onset of the positive feedback. Then, the Péclet number in the applicable shear force equation is

$$Pe = \frac{\eta M \phi \sin \omega}{kA} \quad (3.140)$$

where ϕ is the bulk velocity in the primary direction. If we assume that the cross section is square, the hydraulic diameter is the length of a side ($D = L = V/A$). If we also assume that the area factor is one ($k = 1$), then

$$Pe = m\rho\eta D\phi \sin \omega \quad (3.141)$$

The factor $m\rho\eta D\phi$ is the Reynolds number; therefore

$$Pe = Re \sin \omega \quad (3.142)$$

The discretization scheme becomes nearly saturated (as the upwind scheme) at roughly $Pe = 10$ (see Figure 3.8) whereas turbulence begins at roughly 2300 [177]. If we assume that turbulence corresponds to saturation of the discretization scheme, $\omega \approx 0.25^\circ$.

3.7.4 Thermal Advection and Conduction

As mentioned in Section 3.4.4, the thermal transfer through an interface i is divided into advective and diffusive parts:

$$\dot{Q}_i = \dot{Q}_{Ai} + \dot{Q}_{Di} \quad (3.143)$$

This follows the form of Equation 3.71 where the transported quantity (X) is heat (Q). This does not imply that heat is treated as a thermodynamic property; it only appears in a flow rate

or a partial derivative, since it is process-dependent. The advective term includes a component of enthalpy. The diffusive part is thermal conduction.

As mentioned in Section 3.4.4, thermal convection is the serial combination of thermal conduction and the advection of energy. This is shown in Figure 3.13. The thermal conduction typically occurs between a solid and a fluid, which is an exchange process in the model. However, we may assume that the thermal exchange occurs in a region of negligible thickness centered at the solid-fluid interface and that the fluid has the same temperature of the fluid there (i.e, no contact resistance). Then, thermal convection is governed by two remaining processes in series: (1) thermal conduction between the fluid in the negligibly thin region and the fluid in a larger neighboring region and (2) the advection through the larger region in the direction transverse to the boundary.

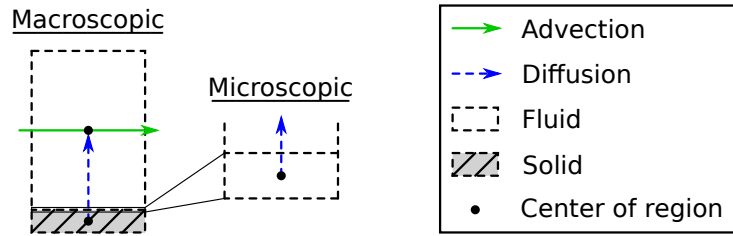


Figure 3.13: Thermal convection in the model.

3.7.4.1 Advection

Energy is advected according to the generalized advective transport equation (3.72) where the property $\partial X/\partial N$ is sT :

$$\dot{Q}_{Ai} = \dot{N}_i (sT)_i \quad (3.144)$$

The factor $(sT)_i$, combined with g_i of the material term, comprises the enthalpy associated with the material flow (as discussed in Sections 3.4.4 and 3.5.3.1). That factor is analogous to $m\phi_i$ in the equations for translational advection (3.121 and 3.126). However, whereas the specific mass at the boundary is equal to the specific mass in the region, the specific entropy at the boundary is not generally equal to the specific entropy in the region ($m_i = m$, but $s_i \neq s$). The temperature (T) is the diffusion-driving property, like the velocity (ϕ).

3.7.4.2 Diffusion

Like friction, thermal conduction or diffusion follows the form of the general transport equation (3.75) with an adjustment factor (here, Nu_Q):

$$\theta \dot{Q}_i = \frac{Nu_Q k_i A_i}{L_i} (T_i - T) \left(1 + e^{\mp \theta C_v \phi_{\perp i} / 2k_i A_i} \right) \quad (3.145)$$

The variable T_i is the temperature at boundary i , and \dot{Q}_i is the rate of thermal conduction into the region through the boundary. The temperature in the region (T) may be considered known, since it is a state of the energy balance (Section 3.4.4).

The thermal resistivity, θ , can be estimated from the generalized definition of resistivity (Equation 3.78).

$$\theta = \frac{3\tau}{\lambda^2 \rho c_v} \quad (3.146)$$

where the partial derivative is taken at constant volume because the regions have fixed volume. This equation matches the thermal conductivity given by Present [148], noting that thermal conductivity is the reciprocal of thermal resistivity [172].

The adjustment factor Nu_Q is the traditional (thermal) Nusselt number. It allows the thermal conduction equation to be used at relatively coarse levels of discretization. It is the ratio of the average inward temperature gradient along the perimeter divided by the difference between the boundary temperature and the bulk temperature over the distance from the boundary to the center of the region:

$$Nu_Q \equiv \left(\frac{\partial T}{\partial x} \right)_i \bigg/ 2 \frac{T_i - T}{L_x} \quad (3.147)$$

At a macroscopic level, the temperature profile is not linear under thermal convection because the conducted heat is carried away by advection transverse to the boundary (Figure 3.13).

Here, the Nusselt number is defined more generally than usual [172]; it need not be used at a solid boundary. Nonetheless, if it is applied to internal flow (fluid bounded by a solid conduit), its values correspond to the traditional ones. For fully developed laminar flow in a circular pipe under a uniform boundary temperature, the Nusselt number is approximately 3.66. With uniform heat flux, it is 48/11 or approximately 4.36 [172].

3.7.4.3 Discussion

If the bulk velocity (and advective current) is zero along the axis of transport and the area factor and Nusselt number are unity,^{xv} then the thermal conduction equation (3.145) reduces to

$$\theta \dot{Q}_i = \frac{2A_i}{L_i} (T_i - T) \quad (3.148)$$

If the temperature gradient is uniform, energy is not stored due to diffusion ($\dot{Q}_n = -\dot{Q}_p = AJ$).

In that case, we can combine the negative- and positive-side equations as

$$\theta J = \frac{T_n - T_p}{L} \quad (3.149)$$

which is the first-order approximation to Fourier's law or the law of heat conduction [155,172].

The thermal Nusselt number is related to the heat transfer coefficient by

$$Nu_Q = \frac{h\theta L}{1 + e^{\mp\theta N c_v \phi_{\perp}/2kA}} \quad (3.150)$$

where $L/(1 + e^{\mp\theta N c_v \phi_{\perp}/2kA})$ is the characteristic length. Substituting this into the thermal conduction equation (3.145) under the assumption that the area factor is unity ($k = 1$),

$$\dot{Q}_i = hA_i (T_i - T) \quad (3.151)$$

which is Newton's law of cooling [172].

3.8 Transport Properties

Highlights:

- By default, the model estimates all diffusion properties based on the kinetic theory of gases, so they only depend on temperature, pressure, and the particle's mass and diameter. It includes refinements where they are available and necessary.

Due to the assumptions implicit in the diffusive transport equation (3.74), the equations for the dynamic compressibility (3.124), material resistivity (3.107), fluidity (3.128), and thermal

^{xv}The latter could be due to the absence of advection or a fine level of discretization, either of which makes a linear temperature profile reasonable within the region.

resistivity (3.146) are only taken to be estimates. However, they are useful if more precise data is not available. Those equations require the collision interval (τ) and the mean free path (λ), which were given for the exchange properties (Section 3.6) under additional assumptions from the kinetic theory of gases.

The estimate of fluidity (Equation 3.128) predicts that fluidity is independent of pressure or specific volume, which accurately matches observations [148]. However, the fluidity generally falls more rapidly with temperature than predicted [148]. Higher accuracy can be achieved for many common gases using the correlations by Svehla, McBride, and Gordon [178, 179]. Those correlations have the following form:

$$\eta = b_{\zeta 1} \ln T + b_{\zeta 2}/T + b_{\zeta 3}/T^2 + b_{\zeta 4} \quad (3.152)$$

Higher accuracy can be achieved for thermal resistivity than Equation 3.146 using correlations from the same source [178, 179]:

$$\theta = b_{\theta 1} \ln T + b_{\theta 2}/T + b_{\theta 3}/T^2 + b_{\theta 4} \quad (3.153)$$

3.9 Electrochemical Reactions

Highlights:

- The model uses the Butler-Volmer equation, which is derived in this section.
- The charge transfer coefficient appears explicitly in this derivation.

The electrochemical reactions are described using the Butler-Volmer equation, but we will first derive it from the exchange equations of the model (Section 3.5). This derivation (1) relates the exchange current of the Butler-Volmer equation to the previously established parameters and (2) indicates how to partition the reaction equations into submodels for the implementation in Chapter 4.

3.9.1 Context

The electrochemical reactions of a proton exchange membrane fuel cell (PEMFC) occur near the interface of the electrode or catalyst (e.g., platinum), which has free (conductance-band)

electrons, and the electrolyte or ionomer, which has free protons.^{xvi} The interface is located in the catalyst layer as shown in Figure 3.14. A thin layer of charge exists just inside the surface of the catalyst due to an excess or deficit of electrons there. This charge attracts or repels positive ions from the neighboring ionomer. We will assume that these ions are the protons themselves, but they may be a combination of other intermediaries of the anodic or cathodic half reaction (Equations HOR and ORR). Over the electrolytic double layer (EDL), the charge of the excess (or deficit) electrons balances the charge of the excess (or deficit) protons.

The electrolytic side of the double layer consists of a surface layer and a diffuse layer. The protons are the most concentrated or depleted (depending on attraction or repulsion) at the surface layer which is offset from the interface by a small gap (due to the nonzero particle size). Over the diffuse layer, the concentration (or depletion) decays with distance from the interface much more slowly than for electrons.

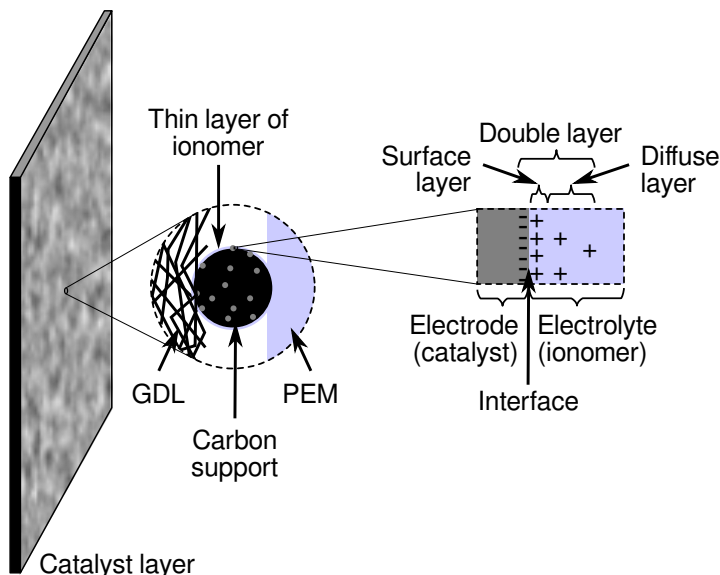


Figure 3.14: The electrochemical reaction occurs in the double-layer region near the catalyst/ionomer interface (based on [17, p. 74] and [81]).

The charge difference leads to an electric field which exerts forces on the ions, as shown in Figure 3.15. For simplicity, we will apply assumptions (such as sheet charge and others detailed below) that result in a uniform electric field and uniform drift velocities over the gap. Although

^{xvi} The term “protons” is used for simplicity, but to be more precise, the charge carriers are hydrogen nuclei.

the ions drift (i.e., are advected) across the gap, there is no net transport because advection cancels diffusion. We assume that (1) the ions are reacted or stored as they pass the edge of the gap, (2) the ionomer does not conduct electrons, and (3) the catalyst does not conduct protons.

The reaction is driven by a difference in the concentrations of electrons between the free (conductance-band) and bound (e.g., as H relative to H⁺) forms at the edges of the gap. Those concentrations are each a consequence of the constraint that advection cancels diffusion, beginning from the interface where we assume that the positive and negative ions exist at their bulk or charge-neutral concentration.^{xvii}

Although the reaction coordinate is normal to the surface, the surface of the catalyst is oriented in all global directions. Therefore, we will consider the reaction to be an exchange process. The EDL is considered a distinct phase that only occupies part of the region.

In summary, the reaction occurs within the double layer which contains a non-conducting gap surrounding the electrode/electrolyte interface. There is potential for a charge difference across the double layer which produces an electric field that causes ions to drift across a non-conducting gap. We assume that the ions drift to the edge of the gap and are stored or reacted there. The reaction is a diffusive process driven by the difference in the concentration of electrons between the edges of the gap. Within the gap, there is no net transport; drift must cancel diffusion. This requirement establishes the concentrations at the edges of the gap, which yields the reaction rate.

3.9.2 Equations

We will assume that the limiting reaction is the combination or separation of the transported ions. In the context of the PEMFC, this is



We will return to this reaction below. We will assume that the following reaction is in equilibrium and thus does not reduce the rate of the anodic reaction:



^{xvii}At this concentration, the charge of the ion is balanced by the charge of the substrate. For example, the e⁻ and Pt⁺ would have the same concentration or H⁺ and SO₃⁻ would have the same concentration.

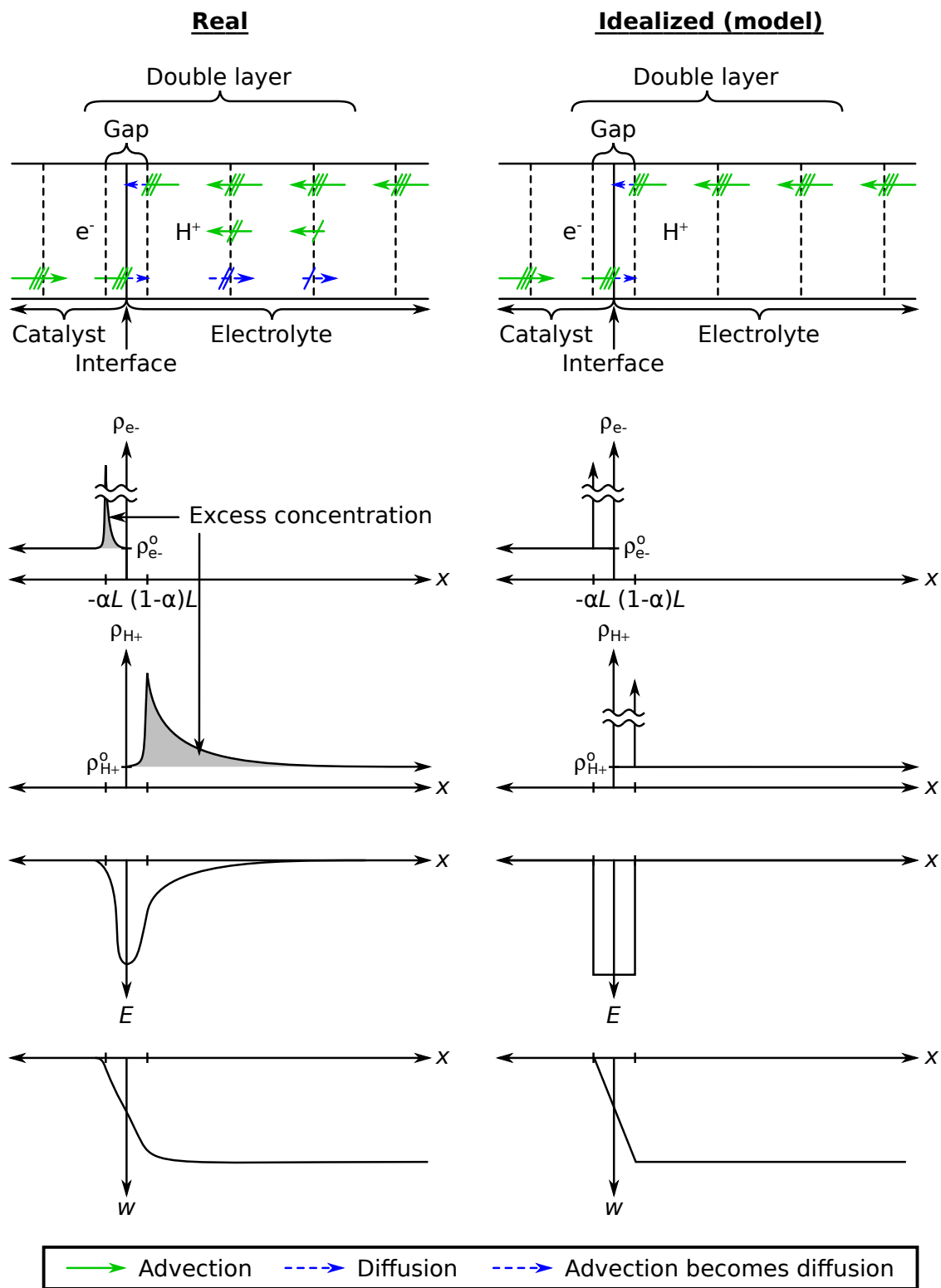


Figure 3.15: Advection, diffusion, concentration, electric field, and electrical potential along the reaction coordinate.

In the cathode, we will assume that the following reaction is in equilibrium:



We will also assume that the water is vapor. This assumption does not have a large effect because the evaporation/condensation process is rapid (see Section 3.5.1). At equilibrium, the stoichiometric sum of the specific Gibbs energies of the products and reactants is zero. Therefore, from the previous two equations, the specific Gibbs energy of the hydrogen (H) is $\frac{1}{2}g_{\text{H}_2}$ in the anode and $\frac{1}{2}g_{\text{H}_2\text{O}} - \frac{1}{4}g_{\text{O}_2}$ in the cathode.

Like phase change, a reaction is driven by concentration differences. The rate of the electrochemical reaction is the rate of diffusive exchange of electrons. It follows from Equation 3.36 for material ($X = N$, $\gamma = \rho$, $\partial X/\partial \gamma = V$) that

$$\frac{8}{3\pi} \tau_{e^-} \dot{N}_{\text{D}ie^-} = k_{ie^-} V_{e^-} (\rho_i - \rho_{e^-}) \quad (3.157)$$

The conversion concentration, ρ_i , is the product of a rate constant and the concentrations of the other species (besides electrons) each raised to the power of their stoichiometric coefficients: $b\rho_{\text{H}}/\rho_{\text{H}^+}$. We will assume that state of the hydrogen (H) varies isothermally and as an ideal gas up to the specific Gibbs energy of conversion (g_{H}). Therefore,

$$\frac{8}{3\pi} \tau_{e^-} \dot{N}_{\text{D}ie^-} = k_{ie^-} V_{e^-} \left(\frac{bp^0}{\rho_{\text{H}^+} T} e^{g_{\text{H}}/T} - \rho_{e^-} \right) \quad (3.158)$$

where p^0 is the reference pressure of the ideal gas.

The concentrations of the electrons and the protons are evaluated at the edges of the gap. We can determine those concentrations from the transport model under two assumptions: (1) at the interface, the concentrations are equal to the bulk concentrations and (2) across the gap, there is no net transfer of ions. The second assumption implies that advection cancels diffusion. If two adjacent regions have the same resistance to diffusion of the ion, it follows from Equations 3.97 and 3.104 that

$$0 = A\phi\rho + \frac{\rho_1 - \rho_2}{R} \quad (3.159)$$

where ϕ is the velocity at a boundary and ρ is the excess concentration there. The resistance (R) is $\zeta L/kA$ (see Equation 3.76, where the resistivity r is the material resistivity ζ), but we will

assume that the area factor (k) is one. Therefore,

$$0 = \zeta \phi \rho + \frac{\rho_1 - \rho_2}{L} \quad (3.160)$$

We can write this as a differential equation (taking the limit as $L \rightarrow 0$):

$$\zeta \phi \rho dx = d\rho \quad (3.161)$$

From the interface to the edge of the gap, the solution for the first ion is

$$\rho_1 = \rho_1^o e^{-\alpha \zeta_1 \phi_1 L} \quad (3.162)$$

where α is the fraction of the length of the gap (L) that exists on the side of the first ion. It is commonly called the charge transfer coefficient [180]. The factor of $\zeta \phi L$ in the argument of the exponential is the Péclet number of diffusive material transport (Section 3.7.1.2). The variable ρ_1^o is the bulk or charge-neutral concentration of the ion. The solution for the second ion is

$$\rho_2 = \rho_2^o e^{(1-\alpha)\zeta_2 \phi_2 L} \quad (3.163)$$

The velocities in the previous two equations can be determined from the momentum balance (Equation 3.190) under the assumption that the velocity is at steady state and there is no gravitational acceleration, surface force, or force due to advective exchange.

$$ZE = m\dot{\Phi}_{DE} \quad (3.164)$$

We will assume that the electric field is uniform across the gap ($E = w/L$). The drag force ($m\dot{\Phi}_{DE}$) is $N\phi/\mu$ due to Equation 3.55 under the assumption that the drag is with a stationary interface (zero mediation velocity) and the adjustment factor is one. The mobility (μ) is $1/\zeta T$ according to the Einstein relation (Section A.5). Therefore,

$$\phi = \frac{zE}{\zeta T} \quad (3.165)$$

With this, the concentrations of the two ions are

$$\rho_1 = \rho_1^o e^{-\alpha z_1 w/T} \quad (3.166)$$

and

$$\rho_2 = \rho_2^o e^{(1-\alpha)z_2 w/T} \quad (3.167)$$

Therefore, the reaction rate (from Equation 3.158) is

$$\frac{8}{3\pi} \tau_{e^-} \dot{N}_{Die^-} = k_{ie^-} V_{e^-} \left(\frac{b p^o}{\rho_{H^+}^o T} e^{[g_H \pm (\alpha-1)w]/T} - \rho_{e^-}^o e^{\pm \alpha w/T} \right) \quad (3.168)$$

where \pm is positive if the electrons are on the first side (and protons on the second) or negative if the protons are on the first side (and electrons on the second).

As written, the previous equation may lead to numerical issues because the arguments of the exponentials may be large at open circuit (on the order of 40 in the cathode of a PEMFC). To help, we will use a relative electrical potential—the activation overpotential—instead of the double-layer potential. The overpotential is relative to the equilibrium potential:

$$\boxed{w' = w - w_{eq}} \quad (3.169)$$

At the equilibrium potential (w_{eq}), there is no net reaction ($\dot{N}_{Die^-} = 0$). From Equation 3.168,

$$\pm w_{eq} = g_H + T \ln \left(\frac{b p^o}{\rho_{e^-}^o \rho_{H^+}^o T} \right) \quad (3.170)$$

By default, the logarithmic term is removed in the model, which implies that $b = \rho_{e^-}^o \rho_{H^+}^o T / p^o$.

$$\boxed{\pm w_{eq} = g_H} \quad (3.171)$$

In terms of the overpotential, Equation 3.168 is

$$\frac{8}{3\pi} \tau_{e^-} \dot{N}_{Die^-} = k_{ie^-} V_{e^-} \left(\frac{b p^o}{\rho_{H^+}^o T} \right)^\alpha (\rho_{e^-}^o)^{1-\alpha} e^{g_H/T} \left[e^{\pm(\alpha-1)w'/T} - e^{\pm \alpha w'/T} \right] \quad (3.172)$$

or

$$\boxed{\dot{N}_{Die^-} = I^o \left[e^{\pm(\alpha-1)w'/T} - e^{\pm \alpha w'/T} \right]} \quad (3.173)$$

where the exchange current, I^o , is defined below. Due to this equation, heat is generated at the rate of $\mp w' \dot{N}_{Die^-}$ in the gap. We can write the reaction rate in terms of electrical current from the first side towards the second ($zI = \mp \dot{N}_{Die^-}$):

$$zI = \pm I^o \left[e^{\pm \alpha w'/T} - e^{\pm(\alpha-1)w'/T} \right] \quad (3.174)$$

This is the Butler-Volmer equation [170, 181, 182], where zI is the electrical current. Again, \pm is positive if the electrons are on the first side or negative if the protons are on the first side.

The exchange current, I^o , is

$$I^o = \frac{3\pi}{8} \frac{k_{ie^-}}{\tau_{e^-}} V_{e^-} \left(\frac{bp^o}{\rho_{H^+}^o T} \right)^\alpha (\rho_{e^-}^o)^{1-\alpha} e^{g_H/T} \quad (3.175)$$

Since it is assumed that $b = \rho_{e^-}^o \rho_{H^+}^o T / p^o$,

$$I^o = \frac{3\pi}{8} \frac{k_{ie^-}}{\tau_{e^-}} (V_{e^-} \rho_{e^-}^o) e^{g_H/T} \quad (3.176)$$

The factor of $V_{e^-} \rho_{e^-}^o$ is the number of electrons that would exist in the gap if they existed at the bulk concentration over the entire gap (either as free e^- or as H in excess of H^+). In practice, the exchange current (or exchange current density) is an empirical, tunable parameter or correlation (e.g., [88]).

The electrical potential across the gap, w , depends on the charge in the EDL, its distribution, and the permittivity of the EDL. We have assumed that the charges are uniformly distributed over parallel planes (i.e., infinitely thin sheet charges) and the electric field is uniform over the gap. Assuming also that the electrical capacitance is constant, the following dynamic relationship is introduced:

$$zI = \frac{\varepsilon A}{L} \frac{\partial w}{\partial t} \quad (3.177)$$

where ε is the electrical permittivity over the gap and $\varepsilon A/L$ is the electrical capacitance. It is important to note that the capacitance is associated with the electrical potential (w), not the overpotential (w'). The total current is the sum of the storage current from this equation and the reaction current from Equation 3.174.

The force applied to ion i at the edge of the gap is

$$m \dot{\Phi}_D = \mp z_i \rho_i^o A w \quad (3.178)$$

This force that yields the drift velocity within the gap. The electrical capacitor is the only component of the entire model (this chapter) that does not conserve translational momentum. The forces on the positive and negative ions will not be equal and opposite if the ions have different bulk concentrations (ρ_i^o).

3.10 Detailed Conservation Equations

Highlights:

- The conservation equations explicitly and exactly account for all flows due to exchange and transport. Convergence does not depend on the mesh size as in computational fluid dynamics (CFD) methods [183] because the flow rates are shared between adjacent regions. As long as the simulation runs, conservation is guaranteed within the simulation tolerance. This allows the discretization to be coarse; only several regions are necessary to model a PEMFC.
- The model includes all of the effects in the compressible Navier-Stokes equations—convective acceleration, unsteady acceleration, pressure gradients, shear stresses, and body force.
- The energy conservation equation includes heat generation due to all of the modeled diffusive or irreversible effects.
- Thermal advection is essential to phase change and reactions; it captures the enthalpy of formation. Translational advection may be excluded in the model of a typical PEMFC since the surface layer is stationary. However, it is included for completeness and may be important in other devices.

The basic conservation equations were given in Section 3.4. Here they are expanded with the terms developed in the exchange, transport, and reaction sections (3.5, 3.7, and 3.9) and further analyzed.

The analysis includes the consideration of time constants. The time constants of the various processes are given by

$$\tau = \left(\frac{\partial X}{\partial \gamma} \right) \left(\frac{\partial \gamma}{\partial \dot{X}} \right) \quad (3.179)$$

where X is a transferred quantity and γ is an intensive property that drives its diffusive exchange or transport. Thus, each time constant is the product of a generalized capacitance ($\partial X / \partial \gamma$) and a generalized resistance ($\partial \gamma / \partial \dot{X}$)

3.10.1 Material

The intake terms of the basic material conservation equation (Equation 3.25) can be divided into exchange and transport.

$$\underbrace{\frac{\partial N}{\partial t}}_{\text{storage}} = \underbrace{\sum_{i \in E} \dot{N}_{Di}}_{\text{exchange}} + \underbrace{\sum_{i \in T} (\underbrace{\dot{N}_{Di}}_{\text{diffusion}} \pm \underbrace{\phi_{\perp i} \rho_i A_i}_{\text{advection}})}_{\text{transport}} \quad (3.180)$$

where the transport current has been expanded into advective and diffusive parts using Equations 3.104 and 3.105. As noted in Section 3.5, there is no advective exchange of material. The sum in the exchange group is over all phase change and reaction processes in which the configuration participates. The sum in the transport group is over the boundaries.

If the phase in which the configuration exists has constant volume, the material conservation equation may be written as

$$V \frac{\partial \rho}{\partial t} = \sum_{i \in E} \dot{N}_{Di} + \sum_{i \in T} (\dot{N}_{Di} \pm \phi_{\perp i} \rho_i A_i) \quad (3.181)$$

The advective currents ($\phi_{\perp i} \rho_i$) may be written as J_i . Dividing this by volume, expanding the transport terms, and assuming that there is no diffusive transport,

$$\frac{\partial \rho}{\partial t} = \frac{\sum_{i \in E} \dot{N}_{Di}}{V} + \frac{J_{xn} - J_{xp}}{L_x} + \frac{J_{yn} - J_{yp}}{L_y} + \frac{J_{zn} - J_{zp}}{L_z} \quad (3.182)$$

which is the first-order spatial approximation of the continuity equation [154, 172]. Nevertheless, a first-order approximation is sufficient because the model's formulation allows the temporal integrator to guarantee convergence within a prescribed simulation tolerance. The fluxes or currents are represented by variables that are explicitly shared between adjacent regions; therefore, the mass lost by one region is exactly the mass gained by another. This is the essence of the continuity equation.

3.10.1.1 Time Constants

Based on Equation 3.179, the time constant due to material exchange is

$$\tau_{NE} = V \frac{\partial \rho}{\partial \dot{N}} \quad (3.183)$$

This implies that the associated time constants are zero for isochoric species. For a gas undergoing isothermal phase change, the time constant is

$$\tau_{NE} = \tau' \frac{N_g T_c}{N_c} \left(\frac{\partial p_g}{\partial \rho_g} \right)_T \exp \left(\frac{g_c - g_g}{T_c} \right) \quad (3.184)$$

based on Equations 3.42 and 3.14, where τ' is the effective collision interval (Equation 3.43). The subscript g indicates the gas phase and the subscript c indicates the condensed phase. The partial derivative $\frac{\partial p}{\partial \rho}$ is $-\nu^2 \frac{\partial p}{\partial \nu}$, which can be determined using Equation 3.4 or 3.6. If the gas is ideal, then

$$\tau_{NE} = \tau' \frac{N_g T_c}{N_c T_g} \exp \left(\frac{g_c - g_g}{T_c} \right) \quad (3.185)$$

For a species i participating in an isothermal electrochemical reaction, the time constant is

$$\tau_{NE} = \frac{2N_i T}{J^0 A (e^{Pe/2} + e^{-Pe/2})} \left(\frac{\partial p_i}{\partial p_i} \right)_T \quad (3.186)$$

based on Equations 3.173 and 3.14, assuming the exchange current density (J^0) is constant and the charge transfer coefficient (α) is one half. Again, the partial derivative $\partial p / \partial \rho$ can be determined using Equation 3.6. If the species is an ideal gas, then

$$\tau_{NE} = \frac{2N_i}{J^0 A (e^{Pe/2} + e^{-Pe/2})} \quad (3.187)$$

For an electrical species,

$$\tau_{NE} = \frac{2T \varepsilon}{J^0 L (e^{Pe/2} + e^{-Pe/2})} \quad (3.188)$$

assuming that the only storage of the charge carrier is in the electrical capacitor with permittivity ε described by Equation 3.177.

Using the material diffusion equation (3.106), the time constant due to material transport through a boundary at constant extensive volume is

$$\tau_{NT} = \frac{\zeta V L}{kA (1 + e^{\mp \zeta V \phi_{\perp} / 2kA})} \quad (3.189)$$

where the volume (V) is the volume of the phase. However, the area (A) is the cross-sectional area of the region along the axis of transport and L is the length of the same. If the bulk

material flow is towards a boundary, the associated time constant is smaller because the effective transport length is smaller. The limiting cases are $\tau_{NT} = 0$ (direct coupling) for infinite dimensionless material flow rate towards a boundary, $\tau_{NT} = \zeta L^2/2k$ for pure diffusion, and $\tau_{NT} = \zeta L^2/k$ for infinite flow away from the boundary.

The previous equation and similar ones below are boxed because they are implemented in the model (Chapter 4). However, they are nonessential; they are only used to characterize the behavior due to other properties.

3.10.2 Translational Momentum

The conservation of translational momentum (Equation 3.28) can be written in terms of an intensive derivative by expanding the transient term using the chain rule and incorporating the conservation of material (Equation 3.25):

$$\boxed{M \frac{\partial \phi}{\partial t} + Ma + ZE + A\Delta p_i = \sum [m(\phi_i - \phi)\dot{N}_i + m\dot{\Phi}_{Di}]} \quad (3.190)$$

For each transport interaction, the current consists of advective and diffusive parts according to Equation 3.104. For each advective exchange interaction, the current (\dot{N}_i) is only the diffusive current (\dot{N}_{Di}), since there is no advective material exchange.

The previous equation can be expanded by separating the exchange and transport terms and applying Equation 3.48 for advective exchange.

$$M \frac{\partial \phi}{\partial t} + Ma + ZE + A\Delta p_i = \sum_{i \in \text{exchange}} \left[m \begin{cases} \phi_i - \phi & \text{if } \dot{N}_i > 0, \\ 0 & \text{if } \dot{N}_i \leq 0 \end{cases} \dot{N}_i + m\dot{\Phi}_{Di} \right] + \sum_{i \in \text{transport}} [m(\phi_i - \phi)\dot{N}_i + m\dot{\Phi}_{Di}] \quad (3.191)$$

where the first sum is over the exchange interfaces and the second sum is over the transport interfaces. As mentioned in Section 3.5, the exchange interactions are either (1) advective, whereby the diffusion term ($m\dot{\Phi}_{Di}$) is zero or (2) diffusive, whereby the current \dot{N}_i is zero.

This form of the translational momentum balance more clearly shows the conditions that cause the material to accelerate in the region from a Eulerian perspective.^{xviii} The advective term on the second line of Equation 3.191 relates to the time-independent term of the material derivative of translational momentum [168], the convective acceleration term in the Navier-Stokes equations, and as discussed in Section 3.7.2.1, the dynamic pressure. Phase change and reaction have effects if the configuration is produced at a conversion velocity that is different from the present velocity. The conversion velocity depends on the velocities of the configurations being consumed (see Section 3.5.2). If the configuration itself is consumed, the phase change or reaction has no effect since the configuration is consumed at its own velocity. If material enters through a boundary ($\dot{N}_i > 0$) with a velocity greater than the velocity of the material within the region ($\phi_i > \phi$), the velocity of the material in the region will tend to increase. The velocity is also affected by other configurations in the same region and the same configuration in other regions through the diffusion terms ($m\dot{\Phi}_{Di}$). Finally, the body forces (Ma and ZE) and the thermodynamic force ($A\Delta p_i$) affect the velocity.

The thermodynamic pressure at each boundary is a function of the local temperature and concentration ($p_i = p(T_i, \rho_i)$). If the configuration is incompressible, then the thermodynamic pressure is the reference pressure (p^o) and the thermodynamic contribution is zero. The pressure commonly used in the literature (often denoted by p as well) is the sum of the thermodynamic pressure (p) and the nonequilibrium pressure ($\pm m\dot{\Phi}_i/A$) over all the species in a phase.

Equation 3.191 is closely related to the Navier-Stokes equation. If we combine the momentum balances of all the configurations in the region, the exchange terms disappear:

$$M \frac{\partial \phi}{\partial t} + Ma + ZE + A\Delta p_i = \sum_{i \in \text{transport}} [m(\phi_i - \phi)\dot{N}_i + m\dot{\Phi}_{Di}] \quad (3.192)$$

We will assume that concentration is uniform. This implies that there is no material diffusion and each boundary current is directly related to the bulk velocity in the normal direction (e.g.,

^{xviii} Acceleration from a Eulerian perspective (i.e., convective acceleration) indicates that the mean velocity of particles within the region increases over time. This is different from acceleration from a Lagrangian perspective—the acceleration of a particle or fluid parcel.

$\dot{N}_{xn}/V = -\dot{N}_{xp}/V = \rho \phi_x/L_x$). Dividing the equation by volume and applying it to the x axis,

$$m\rho \left(\frac{\partial \phi_x}{\partial t} + \frac{\phi_{xpx} - \phi_{xnx}}{L_x} \phi_x + \frac{\phi_{yypx} - \phi_{ynx}}{L_y} \phi_y + \frac{\phi_{zpx} - \phi_{znx}}{L_z} \phi_z \right) + \rho (ma_x + zE_x) + \frac{P_{xp} - P_{xn}}{L_x} \\ = \frac{\dot{m}\Phi_{Dxnx} + \dot{m}\Phi_{Dxpx} + \dot{m}\Phi_{Dyynx} + \dot{m}\Phi_{Dyypx} + \dot{m}\Phi_{Dznx} + \dot{m}\Phi_{Dzpx}}{V} \quad (3.193)$$

where ϕ_{xnx} is the x component of velocity at the negative-x boundary, ϕ_{yypx} is the x component of velocity at the positive-y boundary, et cetera. The same convention applies to the subscripts in the shear force terms. We can expand the viscous forces using Equation 3.123 for the normal axes and Equation 3.127 for the transverse axes, assuming the Péclet numbers are negligible and the area factors and translational Nusselt numbers are unity. We will also rewrite the volumic body terms ($\rho(ma + zE)$) as $-f/V$:

$$m\rho \left(\frac{\partial \phi_x}{\partial t} + \frac{\phi_{xpx} - \phi_{xnx}}{L_x} \phi_x + \frac{\phi_{yypx} - \phi_{ynx}}{L_y} \phi_y + \frac{\phi_{zpx} - \phi_{znx}}{L_z} \phi_z \right) + \frac{P_{xp} - P_{xn}}{L_x} \\ = \frac{f_x}{V} + 2 \left(\frac{\phi_{xnx} + \phi_{xpx} - 2\phi_x}{\beta L_x^2} + \frac{\phi_{ynx} + \phi_{yypx} - 2\phi_x}{\eta L_y^2} + \frac{\phi_{znx} + \phi_{zpx} - 2\phi_x}{\eta L_z^2} \right) \quad (3.194)$$

If we assume that the dynamic compressibility (β) and the fluidity (η) are both equal to the reciprocal of dynamic viscosity (μ), this is

$$m\rho \left(\frac{\partial \phi_x}{\partial t} + \frac{\phi_{xpx} - \phi_{xnx}}{L_x} \phi_x + \frac{\phi_{yypx} - \phi_{ynx}}{L_y} \phi_y + \frac{\phi_{zpx} - \phi_{znx}}{L_z} \phi_z \right) + \frac{P_{xp} - P_{xn}}{L_x} \\ = \frac{f_x}{V} + 2\mu \left(\frac{\phi_{xnx} + \phi_{xpx} - 2\phi_x}{L_x^2} + \frac{\phi_{ynx} + \phi_{yypx} - 2\phi_x}{L_y^2} + \frac{\phi_{znx} + \phi_{zpx} - 2\phi_x}{L_z^2} \right) \quad (3.195)$$

which is the first-order approximation to the x-axis Navier-Stokes equation for an incompressible, Newtonian, and homogeneous fluid [166, 182, 184]. The first of the two groups on the left side is the material derivative of translational momentum [168]. The variable f is force rather than volumic force. Similar equations apply to the other axes.

Although the model characterizes compressible flow, it is not equivalent to the Navier-Stokes equations for compressible flow. As discussed in Section 3.7.3, the shear force equations are mapped differently than by Stokes' viscous deformation law. In addition, the Navier-Stokes equations consider the effects of compressibility using a volumetric divergence term ($\nabla \cdot \phi$), whereas the model does so with the normal force equation (3.123).

3.10.2.1 Time Constants

Based on Equations 3.179 and 3.55, the time constant due to diffusive translational exchange is

$$\tau_{\Phi E} = \frac{m\mu}{k} \quad (3.196)$$

This time constant is directly related to the mean collision interval (τ) under the assumptions of kinetic theory (Section 3.5). By applying Equation 3.56,

$$\tau_{\Phi E} = \frac{8}{3\pi k} \tau \quad (3.197)$$

which is typically small (assuming $k = 1$)—on the order of 20 ns for oxygen in air (see Section 3.6). Thus, it is appropriate to assume that the velocities of different species are equal within a phase unless they are driven by significant opposing forces. Based on Equations 3.179 and 3.123, the time constant due to dynamic compression is

$$\tau_{\Phi \perp} = \frac{\beta ML}{2kA} \quad (3.198)$$

assuming that the configuration fills the entire region ($V = V_{\text{tot}}$). Based on Equations 3.179 and 3.127, the time constant due to shear force is

$$\tau_{\Phi \parallel} = \frac{\eta ML}{Nu_{\Phi} kA \left(1 + e^{\mp \eta M \phi_{\perp} / 2kA}\right)} \quad (3.199)$$

This time constant depends on the normal velocity at the boundary (ϕ_{\perp}), just as the material transport time constant does (Section 3.10.1.1).

3.10.3 Energy

The energy conservation equation (3.30) can be written in terms of intensive derivatives by expanding the transient terms and incorporating the material conservation equation (3.180):

$$M\phi \frac{\partial \phi}{\partial t} + NT \frac{\partial s}{\partial t} = \sum \left[\left(g_i + (Ts)_i - h + m \frac{\phi_i^2 - \phi^2}{2} \right) \dot{N}_i + \phi_i \dot{m} \Phi_{Di} + \dot{Q}_{Di} \right] \quad (3.200)$$

The Gibbs energy relation (Equation 3.15) has been applied to reduce $g + Ts$ but not $g_i + (Ts)_i$ yet (for reasons that will be apparent). We can eliminate the local acceleration term

(at the expense of adding other terms) by subtracting the translational momentum balance (Equation 3.190) times velocity (ϕ):

$$NT \frac{\partial s}{\partial t} = \phi (Ma + ZE + A\Delta p_i) + \sum \left[\left(g_i + (Ts)_i - h + \frac{m}{2} (\phi_i - \phi)^2 \right) \dot{N}_i + (\phi_i - \phi) m \dot{\Phi}_{Di} + \dot{Q}_{Di} \right] \quad (3.201)$$

Now the interface velocities (ϕ_i) only appear relative to the velocity in the region (ϕ). The new term on the right side, $\phi (Ma + ZE + A\Delta p_i)$, is the opposite (negative) of the rate of energy necessary to overcome the effect of the static forces on the control volume. The advective and diffusive terms (second line) must contribute energy at this rate in order to maintain a steady-state value of specific entropy.

At transport interfaces (boundaries), the material and thermal contributions are considered together; therefore, $g_i + (Ts)_i$ reduces to h_i . At exchange interfaces (transitions) where the configuration is a source (reactant), $(Ts)_i = Ts$ and $\phi_i = \phi$ due to Equations 3.48 and 3.59; therefore, $g_i + (Ts)_i - h + \frac{m}{2}(\phi_i - \phi)^2$ reduces to $g_i - g$. As a result, we can write the energy conservation equation as

$$NT \frac{\partial s}{\partial t} = \phi (Ma + ZE + A\Delta p_i) + \sum_{i \in \text{exchange}} \left[\left\{ \begin{array}{l} g_i + (Ts)_i - h + \frac{m}{2} (\phi_i - \phi)^2 \quad \text{if } \dot{N}_i > 0, \\ g_i - g \quad \text{if } \dot{N}_i \leq 0 \end{array} \right\} \cdot \dot{N}_i + (\phi_i - \phi) m \dot{\Phi}_{Di} + \dot{Q}_{Di} \right] + \sum_{i \in \text{transport}} \left[\left(h_i - h + \frac{m}{2} (\phi_i - \phi)^2 \right) \dot{N}_i + (\phi_i - \phi) m \dot{\Phi}_{Di} + \dot{Q}_{Di} \right] \quad (3.202)$$

where the first sum is over the exchange interfaces and the second sum is over the transport interfaces. The factor $(sT)_i$ in the exchange sum (for products) is the thermal conversion property, which depends on the reactants (sources) according to Equation 3.63. The relation between the specific Gibbs energy at the interface (g_i) and the specific Gibbs energy of the configuration (g) depends on the phase and the rate of the reaction or phase change process. The rate equation

for phase change is implemented within the condensed phase (see Section 3.5.1); therefore, the specific Gibbs energy at the interface of the condensed phase is subject to Equation 3.42. The rate equation for electrochemical reactions is implemented in the phase where electrons are the majority charge carrier (see Section 3.9), and the activation overpotential occurs there. For the other phases, including gas, the specific Gibbs energy at the interface is equal to the specific Gibbs energy of the configuration ($g_i = g$). Those phases do not incur any of the irreversible heat of the process, regardless of the direction of the process.

We can use the previous equation to identify the factors that affect the temperature of the configuration. The temperature will increase if, and only if, the specific entropy increases.^{xix} So far, we have seen that the specific entropy of the configuration will increase if (1) material or translational momentum enters through an interface where the velocity is greater than the velocity of the configuration ($\phi_i > \phi$) or (2) the configuration drives a reaction or phase change due to a difference in specific Gibbs energy. In general, the specific entropy will increase if material enters with a specific enthalpy (h_i) greater than that of the configuration (h). The specific entropy will also increase if heat is conducted into the configuration ($\dot{Q}_{Di} > 0$).

Some of these processes are reversible and some are irreversible. It is possible to identify the irreversible ones by evaluating the net change of extensive entropy in two interacting configurations. However, the transfer mechanisms of the model are clearly delineated as advective and diffusive, and the diffusive processes are irreversible. The advective processes are irreversible only to the extent that they are coupled with diffusion. In Section 3.7, we called this process irreversible advection. In the model, it only occurs in transport—not in exchange.

The energy conservation equation is related to the heat equation. The heat equation is typically written for an entire region, so we add Equation 3.202 over all the configurations in the region. This eliminates the exchange terms. We also assume that there is no material advection or diffusion. Therefore,

$$NT \frac{\partial s}{\partial t} = \phi (Ma + ZE + A\Delta p_i) + \sum_{i \in \text{transport}} [(\phi_i - \phi) m\Phi_{Di} + \dot{Q}_{Di}] \quad (3.203)$$

^{xix}Specific entropy and temperature are related in general by $Tds = c dT$, where T and c are both positive. Based on Equation 3.10, Tds is $c_p^o dT$ if the configuration has no thermal expansion and $c_p^o dT - v dp$ if it is an ideal gas.

We will assume that the pressure is constant; therefore, $T \partial s = c_p \partial T$. Heat is generated by viscous dissipation to the extent that $\sum (\phi_i - \phi) m \dot{\Phi}_{Di}$ exceeds $-\phi (Ma + ZE + A \Delta p_i)$, the rate of energy necessary to overcome the effect of the static forces on the control volume. Therefore,

$$N c_p \frac{\partial T}{\partial t} = \dot{Q}_{\text{gen}} + \sum \dot{Q}_{DTi} \quad (3.204)$$

where \dot{Q}_{gen} is the rate of heat generation and $\sum \dot{Q}_{DTi}$ is the total rate of thermal conduction into the region. We can evaluate the thermal conduction using Equation 3.145 under the assumption that the Nusselt number (Nu_Q) and the area factor (k) are both one. The Péclet numbers are all zero due to the previous assumption that there is no material advection. After dividing by volume,

$$\rho c_p \frac{\partial T}{\partial t} = \frac{2}{\theta} \left[\frac{T_{xn} + T_{xp} - 2T}{L_x^2} + \frac{T_{yn} + T_{yp} - 2T}{L_y^2} + \frac{T_{zn} + T_{zp} - 2T}{L_z^2} \right] + \frac{\dot{Q}_{\text{gen}}}{V} \quad (3.205)$$

This is the first-order spatial approximation to the heat diffusion equation assuming uniform thermal resistivity [172]. The variable T_{xn} is the temperature at the negative-x boundary, T_{yp} is the temperature at the positive-y boundary, et cetera.

3.10.3.1 Time Constants

Based on Equations 3.179 and 3.65, the time constant due to diffusive thermal exchange is

$$\tau_{QE} = \frac{c_p \nu}{k} \quad (3.206)$$

where the specific heat capacity is isobaric (c_p) because the pressures of the configurations are assumed to be at equilibrium (see Section 3.3.2). This time constant is directly related to the mean collision interval (τ) under the assumptions of kinetic theory (Section 3.5). By applying Equation 3.66,

$$\tau_{QE} = \frac{8}{3\pi k} \tau \quad (3.207)$$

which is typically small (assuming $k = 1$)—on the order of 20 ns for oxygen in air (see Section 3.6). Thus, it is appropriate to assume that the temperatures of different species are equal within a phase unless there is significant heat generation due to a reaction, for example. Based

on Equations 3.179 and 3.145, the time constant due to diffusive thermal transport is

$$\tau_{QT} = \frac{\theta C_v L}{Nu_Q k A \left(1 + e^{\mp \theta C_v \phi_{\perp} / 2kA}\right)} \quad (3.208)$$

This time constant depends on the normal velocity at the boundary (ϕ_{\perp}), just as the material transport time constant does (Section 3.10.1.1).

3.11 Summary

This chapter established a multi-component, multi-phase model of advective and diffusive transport and exchange with dynamic conservation of material, momentum, and energy. The model supports electrochemical reactions and phase change between a condensed or absorbed phase and a gas. The model is spatially distributed in three dimensions, yet it has been discretized for differential algebraic equations (DAEs) with exact conservation at every boundary. The next chapter will review the implementation of the model in the Modelica language.

IMPLEMENTATION OF THE MODEL

This chapter describes how the model is implemented in the Modelica language [1]. It serves primarily as a summary for Appendix B, which contains documentation generated from the source code. The documentation contains auto-generated tables, diagrams, and code listings. It also contains discussions and other details such as lists of assumptions which have been manually written and embedded using Modelica's annotations. That information is only referenced here to avoid redundancy.

The introduction below merely sets the context to describe the model library. For an introduction to the Modelica language, see [185] or [15].

4.1 Introduction

The models are special object-oriented classes in the Modelica language. The model library also uses other classes such as connectors, blocks, functions, records, packages, types. The classes are encapsulated and organized hierarchically via inheritance and instantiation. Encapsulation or abstraction hides details about the implementation, leaving only the meaningful characteristics accessible from outside the class [25]. Inheritance creates models or other classes by extending or adding to more general and basic classes. It helps to organize the library and reduce the amount of duplication in the code. Instantiation allows classes to be built by assembling working copies of lower-level classes. This is consistent with the physical hierarchy of the fuel cell (e.g., a cell is an assembly of layers). Figure 4.1 shows that the model is created by instantiating species into phases, phases into subregions, subregions into regions such as a fuel cell layer, and regions into assemblies such as a fuel cell. By convention, the names of classes begin with an uppercase letter and their instances start with a lowercase letter. Subclasses and sub-instances are accessed via dot notation (e.g., `Class.Subclass1.Subclass2`).

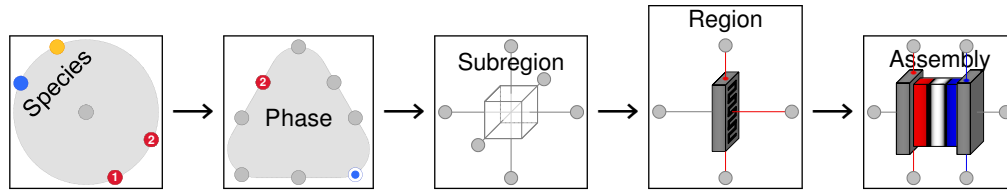


Figure 4.1: Levels of instantiation in the model library (duplicate of Figure 1.8).

Each model is a system with possible subsystems. The physical connectors (i.e., connectors containing efforts and flows, to be discussed later) of the model represent its boundaries. The boundaries may be geometric (e.g., the positive boundary along the x axis; see Figure 3.1) or conceptual (e.g., between two species within a phase).

The equations of a model may be represented graphically or textually. Usually, the higher-level models are graphical whereas the lower-level ones are textual. In the diagrams, lines between physical connectors (Section 4.5) comprise nodes which are subject to the generalized Kirchhoff circuit laws (e.g., efforts are equal and flows sum to zero). The icons represent instances of classes.

The term *state* should be clearly defined because it has two different (but related) meanings in mathematical modeling and thermodynamics. In the context of modeling and Modelica, a state is a scalar time-varying variable of which a derivative is taken.ⁱ The states of a model are necessary and sufficient to determine the values of all other variables of the model at a given time. In the context of thermodynamics, a state is the condition of a system. It encompasses a set of properties with cardinality equal to the degrees of freedom of the system in the sense of Gibbs' phase rule [150,151]. Since the fuel cell models involve thermodynamics, both meanings may be used. A thermodynamic state is represented by a set of model states—generally one for each mode of energy storage (e.g., pressure and temperature).

The remaining sections describe the model library from low to high level. The first sections describe supporting classes such as types (Section 4.2), constants (Section 4.3), records and functions (Section 4.4), and connectors (Section 4.5). The models begin with the subregions in Section 4.9 and continue up to the test models in Section 4.12.

ⁱThese may or may not be the same variables which are wrapped by the `der()` operator since the Modelica translator is usually free to choose appropriate states. Also, those variables may be algebraically coupled, in which case the translator must perform index reduction.

4.2 Quantities

Related section of the documentation:

- B.47 `FCSys.Quantities`

Quantities are types used to represent physical values in the model library. Each quantity has a dimension which is specified in terms of angle (A), length (L), mass (M), particle number (N), and time (T). Quantities may also have default display units and minimum or maximum values.

Instances of quantities are variables. The names of the variables match those in Chapter 3 with the exceptions that (1) Greek letters are spelled in English and (2) subscripts begin with an underscore. Usually, extensive properties are uppercase and intensive properties are lowercase.

The quantities package is described further in the documentation (Section B.47). Throughout the rest of this chapter and the appendix, the `Quantities` package may be abbreviated as `Q` (e.g., `Q.Length`).

4.3 Units

Related sections of the documentation:

- B.72 `FCSys.Units`
- B.73–B.74 `FCSys.Units.*`

The `Units` package is described in Section B.72 with updated information from the related paper [186]. In summary, the approach is based on the concept that a quantity is the product of a number and a unit [187]. The variables in the library represent quantities, in contrast to the usual approach where variables represent numbers or quantities expressed in a unit (where “expressed in” means divided by). When a variable is given a quantity in the library, it is expressed literally as the product of a number and a unit or group of units. This provides convenience and flexibility in entering values (e.g., `101325*Pa` or `1*atm`). When the variable is expressed in a unit, it is divided by that unit (e.g., `p/kPa`).

The units package begins by giving values to certain fundamental, physical, and measurable constants (the speed of light in vacuum and the Rydberg, Josephson, von Klitzing, Faraday, and

gas constants).ⁱⁱ The values are arbitrary except that they can be used to additionally scale the floating point variables. The units are then determined by the accepted values of the constants (e.g., $c = 299,792,458$ m/s from [188]). The constants have independent dimensions such that they are sufficient to determine the values of all other units. Approximately 90 constants, units, and prefixes are defined. All are instances of quantities (as defined in Section 4.2).

As noted in the beginning of Chapter 3, the model equations are written in a system of units where the Faraday and gas are normalized to one. Literally, this means that $1 \text{ mol} = 96,485.3365 \text{ C}$. The coulomb is used as a number of particles just as the mole, but the charge number is applied appropriately when describing an electric field or an electric current. Since the Faraday and gas constants are both normalized, temperature is equivalent to thermal voltage (e.g., $300 \text{ K} \approx 25.85 \text{ mV}$). This also implies that the Boltzmann constant is normalized to q , the particle number representing a single particle. These normalizations simplify the equations and make it straightforward to model electrons and protons like other species (albeit with nonzero charge number).

During the implementation, it was discovered that the document which establishes the International System of Units (SI) [187] may inconsistently define and use angles. Table 3 in that document states that the radian (rad) is defined by $\text{rad} \equiv 1$ and the hertz (Hz) is defined by $\text{Hz} \equiv \text{s}^{-1}$. However, we commonly consider the hertz to be a measure of “cycles per second” or $\text{Hz} = \text{cyc}/\text{s}$, where the cycle (cyc) is defined by $\text{cyc} \equiv 2\pi \text{ rad}$ in trigonometry. Since $\text{rad} \equiv 1$, it follows that $\text{cyc} = 2\pi$. This implies that $\text{Hz} = 2\pi/\text{s}$, which is not consistent with the SI definition ($\text{Hz} \equiv \text{s}^{-1}$).

The discrepancy could be resolved by defining angle as an explicit dimension (like length and time) with units that include the radian, cycle, and degree. Those units are related (e.g., $\text{cyc} \equiv 2\pi \text{ rad}$) but none should be given a fixed value. That is the approach in the Units package, but it is limited by the fixed SI definition of $\text{rad} \equiv 1$. The approach also resolves issues such as the seemingly identical units of energy (J) and torque (Nm) [189] by using angle in the units for torque (e.g., J/rad). In this sense, torque is a unit of energy per swept angle.

ⁱⁱThe radian and candela are also given values to complete the basis. As discussed, the radian must be one due to the definition in [187].

Traditionally, we express the angle in radians but exclude it (since it is one) and then consider torque to be the product of force and radius.

Throughout the rest of this chapter and the appendix, the `Units` package may be abbreviated as `U`. For example, `U.m` is the unit meter.

4.4 Characteristics

Related sections of the documentation:

- B.3–B.30 `FCSys.Characteristics.*`

The `Characteristics` package contains data and functions to correlate physical properties of materials. The data and functions are contained within a package for each chemical species. All of the correlated and derived thermodynamic properties (Sections 3.1 and 3.2) and diffusion properties (Sections 3.5 and 3.7) are coded from the descriptions in the previous chapter. The characteristics are suitable for real and ideal gases, compressible and incompressible fluids, and solids. Table B.3 lists the data and functions.

The `Modelica.media` package (`Modelica.Media [2]`) is not used besides its data. There are four reasons. First, it would be necessary to rewrite the variable declarations since the model library uses a different approach to physical units (see the previous section). Second, many of the functions would need to be wrapped to convert the properties for the equations established in Chapter 3. This would introduce overhead in terms of both computation and software maintenance. Third, some properties are not available in `Modelica.Media`. Other properties and correlations are present but are not needed. Finally, the models are factored differently in the present library. The `Characteristics` package does not include any models or time-varying variables (in the `Species` model instead—next section), unlike `Modelica.Media`.

The virial coefficients (p - v - T relation) for the Leiden (volume-explicit) form are encoded in a matrix (b_v). The power of p/T increases by row and the power of T increases by column, beginning with the powers set by n_v (a 2-tuple). The matrix for the Berlin (pressure-explicit) form (b_p) is computed automatically but is currently limited to the fourth virial coefficient. It can be expanded to an arbitrary order with results from a Computable Document Format

(CDF) file included with the library. The Leiden is directly prescribed instead of the Berlin form so that the specific volume of incompressible species can be entered. The resulting polynomials are encoded in all the property functions using the nested form (e.g., $f(x) = ((\dots + a_{-1-n})/x + a_{-n})/x + a_{1-n} + x \cdot (a_{2-n} + x \cdot (a_{3-n} + \dots))$) for computational efficiency.

The isobaric specific heat capacity-temperature relation also allows arbitrary polynomial order, starting from an arbitrary power. This makes it possible to prescribe constant or temperature-dependent specific heat capacity as needed. The relation is independent of pressure, but following the approach by Dymond et al. [146], the rows of the coefficients matrix (b_c) cover different temperature ranges. An arbitrary number of rows can be included, which is an improvement over `Modelica.Media`.

A base characteristics record is extended for each of the chemical species required for the fuel cell model—positively charged carbon (C^+), negatively charged Nafion sulfonate ($C_{10}HF_{37}O_5S^-$, abbreviated as SO_3^-), electrons (e^-), protons (H^+), hydrogen (H_2), water vapor, liquid water, water absorbed in the ionomer, nitrogen (N_2), and oxygen (O_2). Where available, virial and heat capacity coefficients are included respectively from [146] and [142] (directly or via `Modelica.Media`). These representations are later simplified (e.g., to ideal gas and constant specific heat capacity) as appropriate. The polynomial coefficients for the specific heat capacity of water in the ionomer and the associated integration constants for enthalpy and entropy are set so that the hydration of the ionomer matches the correlation of Springer et al. [64] at 0 and 100% relative humidity. For simplicity, the model is not matched to that correlation over the full range of relative humidity.

4.5 Connectors

Related sections of the documentation:

- B.37 `FCSys.Connectors`
- B.38–B.45 `FCSys.Connectors.*`









Interactions between physical models are described by connections involving pairs of flow and effort variables. The flow variable (or simply “flow”) is typically the rate at which a conserved quantity enters a control volume through the associated interface. The effort variable

(or “effort”) is typically a property which drives diffusion of the quantity. When connectors are joined, a node is formed where the sum of the flows is zero [190] (e.g., Kirchhoff’s current law) and the efforts are equal (e.g., Kirchhoff’s voltage law) [5, 191]. The essence is that connected systems experience the same value of a property at their shared boundary, and when a quantity leaves one system, it immediately enters another (the quantity is not created, destroyed, or stored in the node).

Table 4.1 lists the effort/flow pairs of the physical connectors in the model library. The material pair transfers material between regions. Its flow, \dot{N} , is the current or flow rate of material. The pressure, p , is the thermodynamic pressure. The translational pair transports or exchanges translational momentum due to drag between regions or among species within a region. It is similar to the Modelica mechanical translational connectors (e.g., `Modelica.Mechanics.Translational.Interfaces.Flange_a`) except that the effort is velocity rather than position. The reason is that Modelica mechanics is Lagrangian (i.e., consisting of control masses) whereas the fuel cell model library is Eulerian (i.e., consisting of control volumes). The thermal advective pair exchanges thermal energy between reactants and products in a chemical reaction. The thermal diffusive pair transports or exchanges heat due to thermal conduction between regions or species within a region. It is similar to `Modelica.Thermal.HeatTransfer.Interfaces.HeatPort`. The thermodynamic state is defined at a boundary through temperature and pressure; therefore, sT is known and thermal advection can be determined. The Amagat pair adds the volumes of phases that exist at a certain pressure within a region. The Dalton pair is the opposite; it adds the pressures of species that exist within the volume of a phase. The chemical pair is used for phase change and for the connections leading up to a reaction where material is conserved without reaction. The stoichiometric pair is its opposite. It is used to add the stoichiometrically-weighted chemical potentials of the species involved in a chemical reaction. Its effort is the rate of the reaction, which is common to all of the connected species.

Traditionally, effort/flow pairs are power conjugates (i.e., the product is power), as in bond graphs [192]. However, this is not necessary for energy conservation as long as the variables are sufficient to compute the energy flow rates. Roughly half of the connectors in the Modelica

Table 4.1: Effort/flow pairs of the connectors.

Name	Effort	Flow	Within icon(s)
Material	Pressure p [$\text{ML}^{-1}\text{T}^{-2}$]	Current \dot{N} [NT^{-1}]	
Translational	Velocity ϕ [LT^{-1}]	Force $\dot{m}\phi$ [LMT^{-2}]	
Thermal diffusive	Temperature T [$\text{L}^2\text{MN}^{-1}\text{T}^{-2}$]	Heat flow rate \dot{Q} [L^2MT^{-3}]	
Thermal advective	Temperature times specific entropy Ts [$\text{L}^2\text{MN}^{-1}\text{T}^{-2}$]	Heat flow rate \dot{Q} [L^2MT^{-3}]	
Amagat	Pressure p [$\text{ML}^{-1}\text{T}^{-2}$]	Partial volume V [L^3]	
Dalton	Volume V [L^3]	Partial pressure p [$\text{ML}^{-1}\text{T}^{-2}$]	
Chemical	Chemical potential g [$\text{L}^2\text{MN}^{-1}\text{T}^{-2}$]	Current \dot{N} [NT^{-1}]	
Stoichiometric	Rate of reaction \dot{N} [NT^{-1}]	Net chemical potential g [$\text{L}^2\text{MN}^{-1}\text{T}^{-2}$]	

Standard Library depart from this tradition—namely the mechanical (rotational, translational and multibody), fluid, thermal (heat transfer and fluid heat flow) connectors. In the model, only the translational, electrochemical, and stoichiometric pairs are power conjugated (see Table 4.1).

There are two reasons to use effort/flow pairs that are not power conjugates. When the interaction imposes a static constraint as well as a dynamic one, the variables should be energy conjugates. This is the case for the Modelica mechanical connectors. The power conjugate of force is velocity, but the effort is position instead. The translational and rotational position—not just velocity—of two objects is equal at the point of contact. The Amagat and Dalton pairs are similar in this regard. The Amagat pair is used where the volumes (not just the derivatives of volume) sum to zero. The Dalton pair is used where the pressures (not just the derivatives of pressure) sum to zero.

The second reason to use effort/flow pairs that are not power conjugates is mathematical. In some cases, power conjugation introduces nonlinear model equations. The most common example is thermal conduction. The power conjugate of temperature is entropy flow rate, but heat flow rate is used instead to avoid nonlinear equations that would appear since entropy is not conserved [23, 60]. Considered another way, the power conjugate of heat flow rate is specific entropy, yet specific entropy is an explicit function of temperature (and pressure) rather than vice versa. Also, it is temperature, not specific entropy, that is equal between dissimilar species at a boundary. A similar situation exists with material transport in the model. The power conjugate of pressure is volumetric flow rate, but current is used instead to avoid nonlinear equations that would appear since volume is not conserved (due to compression and mixing). The power conjugate of current is chemical potential, yet chemical potential is an explicit function of pressure (and temperature) rather than vice versa. Also, it is pressure, not chemical potential, that is equal between dissimilar, non-reacting species at a boundary. For chemical exchange, the situation is different. There, chemical potential is a more appropriate connector variable than pressure because chemical potential sums to zero at equilibrium (after stoichiometric weighting). In chemical reactions, we are not interested in force (at least from the macroscopic perspective of the library), so pressure is not necessary.

The connectors have multiple effort/flow pairs. They are organized in a hierarchy as shown in Figure B.4 and discussed in Section B.37. The connector icons (right column in Table 4.1) appear in the model diagrams and icons, usually at the edges. Smaller versions of the connector icons are used for internal connectors (not accessible outside a model). The gray icons represent the boundaries of a cell, region, or subregion. The gold icons represent chemical interactions and the red icons are for inert (i.e., diffusive) exchange. The blue icons represent additivity of pressure and volume.

If an effort/flow pair is in a connector that is disconnected, then the flow will be zero (since the sum of the flows at a node is zero). For example, if a boundary of a region is disconnected, the current, shear force, and rate of thermal conduction is zero.

4.6 Species

Related sections of the documentation:

- B.56–B.70 `FCSys.Species.*`

The `Species` model contains all of the core equations for the exchange, transport, and storage of material, translational momentum, and energy for a single chemical species. Those equations are presented in Sections 3.5, 3.7, and 3.10 of the previous chapter. However, for a robust and manageable implementation, the momentum balances are located at and are normal to the boundaries of a subregion (Section 4.9). Material advection and diffusion are included within the subregion and are not distinguished. The assumptions of the `Species` model (and other details) are listed in Section B.70.

The `Species` model contains numerous parameters, which are listed in Tables B.61 and B.62. Some of the parameters, for instance the geometric dimensions, are common to all of the species within the phase or the subregion. These are fixed (declared `final`) and propagated to the higher level. The remaining parameters are accessible through the parameter dialog of the `Species` instance, as shown in Figure 4.2. Most of the general parameters (Figure 4.2a) pertain to the material characteristics. The instance of the `Characteristic` record (Section 4.4) can be replaced or locally modified. The diffusion properties (μ , ν , ζ , η , and θ) are independently adjustable but default to the functions provided by the chosen `Characteristic`

record. These properties could be given values that depend not only on the thermodynamic state of the species, but also on the velocities (e.g., non-Newtonian fluids) or the properties of other species. The initial conditions (Figure 4.2b) are specified by the type and value of the condition. The assumptions (Figure 4.2c) allow the states to be prescribed (which voids the conservation equations) or the central difference scheme to be used (no upstream discretization).

The thermal and translational Nusselt numbers are implemented as parameters (not time-varying). In theory, it would be possible to model non-Newtonian fluids by allowing the translational Nusselt numbers to depend on shear rate. However, this would introduce nonlinear systems of equations.

The Species model is extended to create Solid and Fluid models. The Solid model is used to represent C^+ and SO_3^- . The Fluid model represents hydrogen, water vapor, liquid water, water in ionomer, nitrogen, and oxygen. The Fluid model is also extended to create an Ion model for charged species (H^+ and e^-). Although this may seem a misnomer (to represent e^- as a fluid), the equations are equivalent to traditional electrical circuit theory (e.g., Ohm's law and the dynamics of self inductance) when electrons are considered to be an incompressible fluid within a solid (the graphite phase). Electrical potential maps to specific Gibbs energy. The only difference is that mobility (μ) is specified in terms of electrical conductivity. Temperature has a slight effect on the potential, but this is consistent with the physics.

The boundary connectors are expanded at each level of the model. At the Subregion level, the connectors of the species are accessed in the form of `phase.species`. For example, the temperature of hydrogen at the negative x-axis boundary of the Subregion model is `xNegative.gas.H2.T`.

General Initialization Assumptions Add modifiers

Component


Name

Comment

Model

Path FCSys.Species.H2.Gas.Fixed

Comment Fixed properties

Icon 

Material properties

Data	<input type="text" value="H2 gas(...)"/>			
μ	<input type="text" value="Data.mu0"/>	cm ² /(v.s)		Mobility
ν	<input type="text" value="Data.nu0"/>	s		Thermal independity
ζ	<input type="text" value="Data.zeta0"/>	N/A		Continuity
η	<input type="text" value="1/(8.96e-6*U.Pa*U.s)"/>	1/(Pa.s)		Fluidity
θ	<input type="text" value="U.m*U.K/(0.183*U.W)"/>	m.K/W		Thermal resistivity

Initialization

phi.start	<input type="checkbox"/>	<input type="text" value="0"/>	cm/s	Velocity
I.start	<input type="checkbox"/>	<input type="text" value="0"/>	A	Current
phi_boundaries.start	<input type="checkbox"/>	<input type="text" value="0"/>	cm/s	Normal velocities at the boundaries
f.start	<input type="checkbox"/>	<input type="text" value="0"/>	N	Total normal translational force on pairs of boundaries
minusDeltaf.start	<input type="checkbox"/>	<input type="text" value="0"/>	N	Dynamic and nonequilibrium compression forces

OK Info Cancel

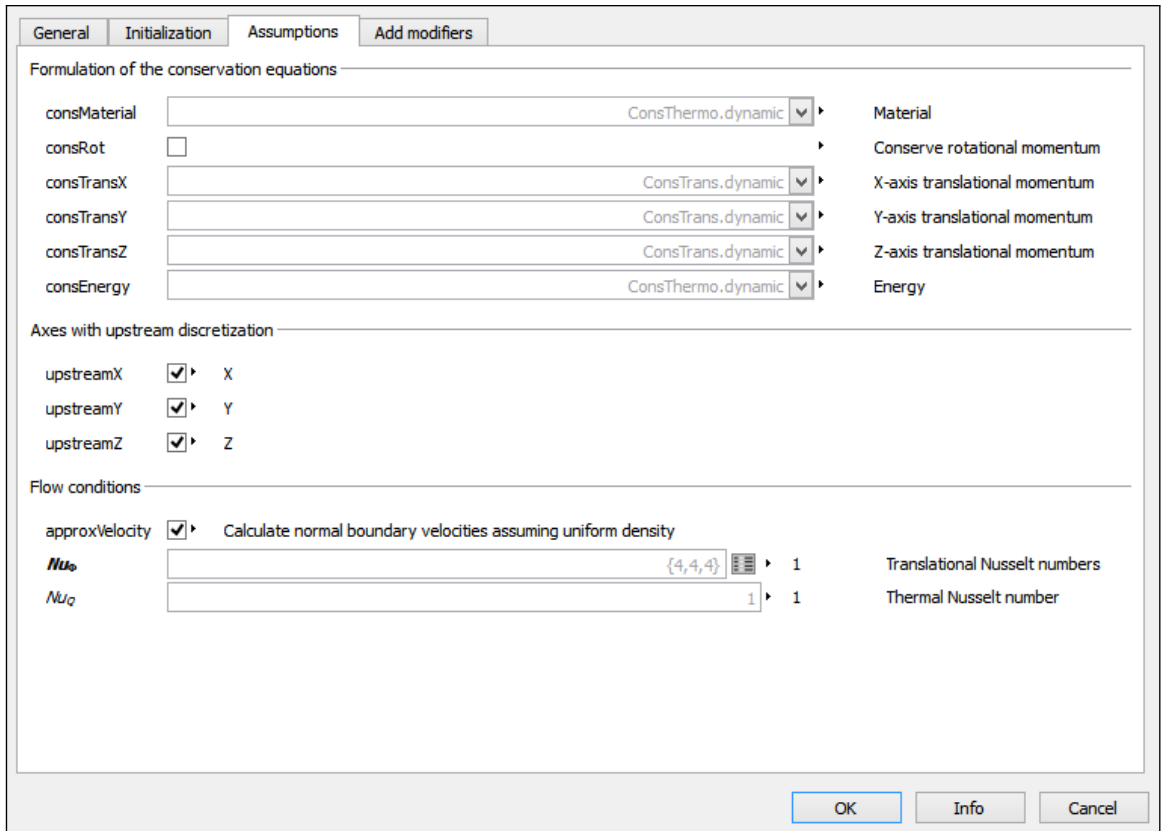
(a) General parameters.

General Initialization Assumptions Add modifiers

initMaterial	<input type="text" value="Init.pressure"/>		Method of initializing the material state
initEnergy	<input type="text" value="Init.temperature"/>		Method of initializing the thermal state
N_{ic}	<input type="text"/>	C	Initial amount of material
ρ_{ic}	<input type="text"/>	C/cc	Initial density
V_{ic}	<input type="text"/>	cc	Initial volume
p_{ic}	<input type="text" value="environment.p_dry"/>	kPa	Initial pressure
T_{ic}	<input type="text"/>	degC	Initial temperature
h_{ic}	<input type="text"/>	kJ/mol	Initial specific enthalpy
g_{ic}	<input type="text"/>	kJ/mol	Initial Gibbs potential

OK Info Cancel

(b) Initialization.



(c) Assumptions.

Figure 4.2: Parameter dialog for the H₂ species.

4.7 Chemistry

Related sections of the documentation:

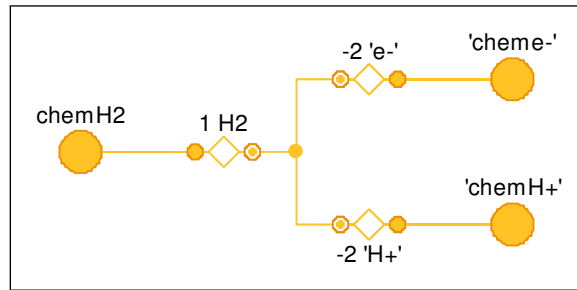
- B.31–B.35 `FCSys.Chemistry.*`

The Chemistry package is a group of models that describe chemical reactions and the affinity between phases. The models are described in the following sections.

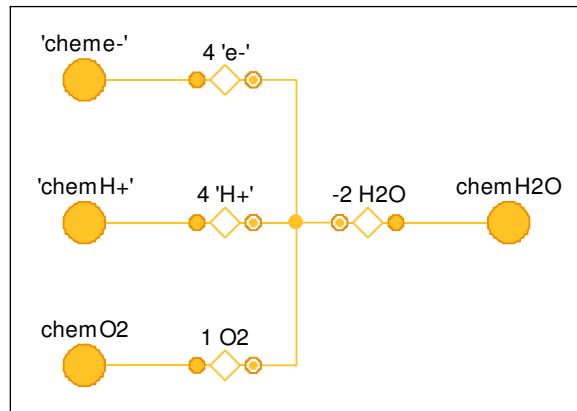
4.7.1 Reactions

The HOR and ORR models establish the chemical equilibria of the hydrogen oxidation and oxygen reduction reactions. Material is transferred through the models according to the stoichiometric relations (Equations HOR and ORR). As material is transferred, it carries translational momentum and energy by advection.

The models are implemented by connecting `ChemicalReaction` adapters for the reactants and products as shown in Figure 4.3. The Modelica code of those adapters, which is listed in Section B.35, expresses the stoichiometric relations and uses Modelica stream operators [193] to handle the advective exchange of momentum and energy according to Equation 3.48 and Equation 3.59. The reaction connectors of those adapters, which have chemical potential as a flow variable (see Table 4.1), are connected to create an internal node that is the point of chemical equilibrium.



(a) HOR ($\text{H}_2 \rightarrow 2\text{e}^- + 2\text{H}^+$).



(b) ORR ($4\text{e}^- + 4\text{H}^+ + \text{O}_2 \rightarrow 2\text{H}_2\text{O}$).

Figure 4.3: Diagrams of the reaction models.

4.7.2 Electron Transfer and Charge Storage

The `ElectronTransfer` model implements the Butler-Volmer equation (3.173). It includes advection and heat generation. The heat is rejected to an `Inert` connector which is connected to the substrate (C^+). The Nernst equation is not implemented in this model (or any other model in the library) because the open circuit voltage is inherent in the properties of the species and their interconnection.

The `DoubleLayer` model stores and releases energy due to charge displacement across the electrolytic double layer. Even though it is instantiated in the `graphite` phase (Section 4.8), it has its own volume (although small) which is introduced through an `Amagat` connector. The `DoubleLayer` model also has an `Inert` connector for translational and thermal exchange. The electrons exit with either the velocity of that connector (typically zero) or the arrival velocity. The first option, which is the default, implies that heat is generated; it is rejected through the same connector.

The Modelica code and a table of parameters for `ElectronTransfer` and `DoubleLayer` are included in Sections B.34 and B.33.

4.7.3 Capillary Pressure

Surface tension and capillary action are essential to remove liquid water from the cell. The `Capillary` model applies capillary pressure between two `Amagat` connectors according to the Young-Laplace equation [194]. It is instantiated within the `CapillaryVolume` model, where it yields a pressure difference between the liquid and the gas. Since the water vapor is modeled as an ideal gas and the liquid has constant volume, the pressure difference shifts the saturation pressure according to the Kelvin equation [194].

The Modelica code and a table of parameters for the `Capillary` model is included in Section B.31.

4.8 Phases

Related sections of the documentation:

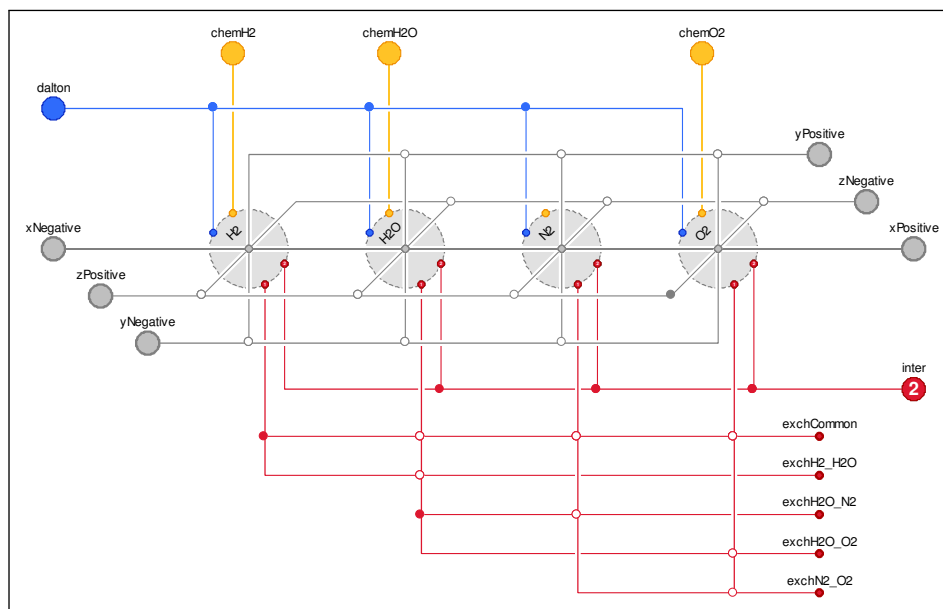
- B.46 `FCSys.Subregions.Phases.PartialPhase`

There are four phase models—one for gas, one for liquid, and two different solids. The `Gas` model contains H_2 , H_2O , N_2 , and O_2 . `Liquid` contains H_2O . `Graphite` contains C^+ and e^- ; `Ionomer` contains H^+ , H_2O , and SO_3^- (abbreviation for $\text{C}_{19}\text{HF}_{37}\text{O}_5\text{S}^-$). The species are conditionally included so that the `Gas` model can be used in both the anode (with N_2 and O_2 removed) and the cathode (with H_2 removed).

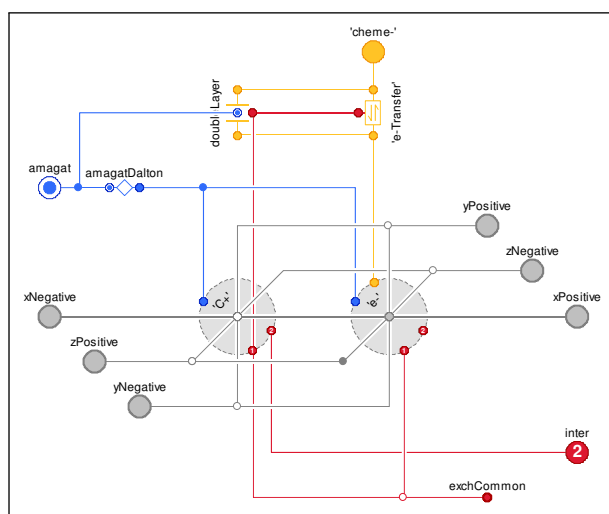
Within a phase, the species are combined according to Dalton's law. The species exert forces on each other and exchange heat according to the diffusive exchange equations (3.55 and 3.65). By setting the independence factors to zero, it is possible to impose the assumption that the temperature or any component of velocity is equal among the species. This causes the translation tool to simplify the model via index reduction.

Figure 4.4 shows the diagrams of the phases. All the phases have boundary bus connectors (`xNegative`, `xPositive`, etc.) to connect to adjacent subregions. Species are labeled by their chemical formula when they are connected to the bus. Each phase also has an `amagatDalton` adapter between Dalton's law (which is applied within the phase) and Amagat's law, which is applied among phases via the `amagat` connector. The phases have `inter` connectors for inter-phase translational and thermal exchange. Internal nodes (e.g., `commonExch`) are included as required for translational and thermal exchange within each phase. Finally, the phases have chemical connectors (e.g., `chemH2O`) for chemical reactions and phase change among the phases. The graphite phase (Figure 4.4b) has a model for electron transfer (`electronTransfer`) and an optional electrolytic double layer capacitance (`doubleLayer`).

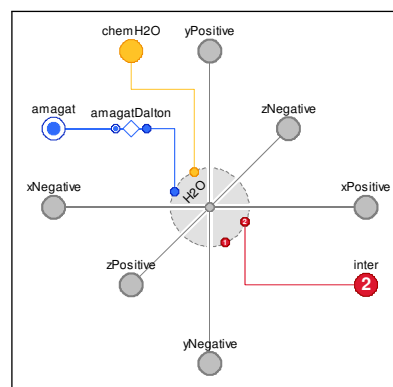
Figure 4.5 shows the parameter dialog for the gas phase. The species can be enabled or disabled and their settings can be changed by opening the sub-dialogs shown in Figure 4.2.



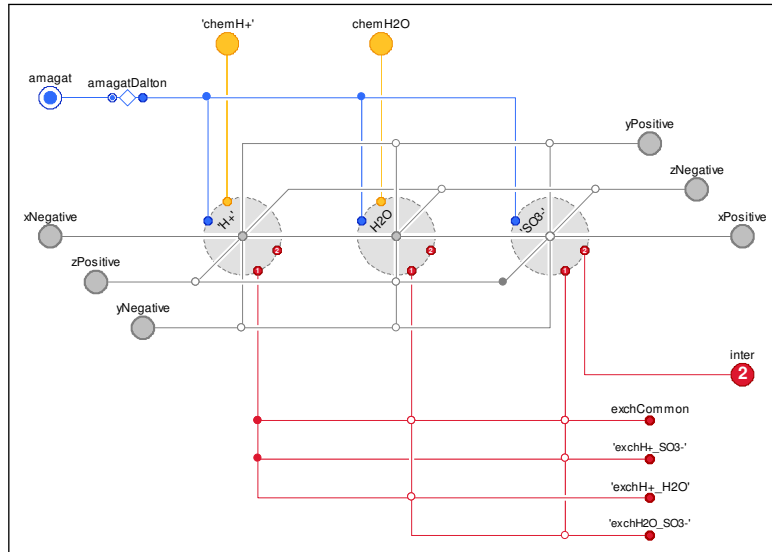
(a) Gas.



(b) Graphite.



(c) Liquid.



(d) Ionomer.

Figure 4.4: Diagrams of the phases.

General Add modifiers

Component

Name

Comment

Model

Path FCSys.Phases.Gas

Comment Gas phase

Icon

Geometry

k Length factors for diffusive transport

Species

Hydrogen (H ₂) <input checked="" type="checkbox"/>	H ₂ model	<input type="text" value="rededare FCSys.Species.H"/>
Water (H ₂ O) <input checked="" type="checkbox"/>	H ₂ O model	<input type="text" value="rededare FCSys.Species.H"/>
Nitrogen (N ₂) <input checked="" type="checkbox"/>	N ₂ model	<input type="text" value="rededare FCSys.Species.N"/>
Oxygen (O ₂) <input checked="" type="checkbox"/>	O ₂ model	<input type="text" value="rededare FCSys.Species.C"/>

Independence factors

common Among species in the phase

OK Info Cancel

Figure 4.5: Parameter dialog for the gas phase.

4.9 Subregions

Related sections of the documentation:

- B.71 `FCSys.Subregions.Subregion`

The smallest unit of spatial discretization is the `Subregion` model, which is shown in Figure 4.6. It contains instances of the phase models—`liquid`, `gas`, `graphite`, and `ionomer`. They are connected to the `volume` model, which imposes the total volume of the subregion and applies capillary pressure between the gas and the liquid (see Section 4.7). All of the phases are connected to the internal `commonExch` node for translational and thermal exchange. The gas and liquid have an additional means of diffusive exchange via the `gasLiq` node. Although not shown in the diagram, the chemical connectors of the phases are connected to the chemical reactions (HOR and ORR) described in Section 4.7. The chemical connectors are directly connected for phase change—between gas and liquid and between gas and the ionomer (also not shown in the diagram).

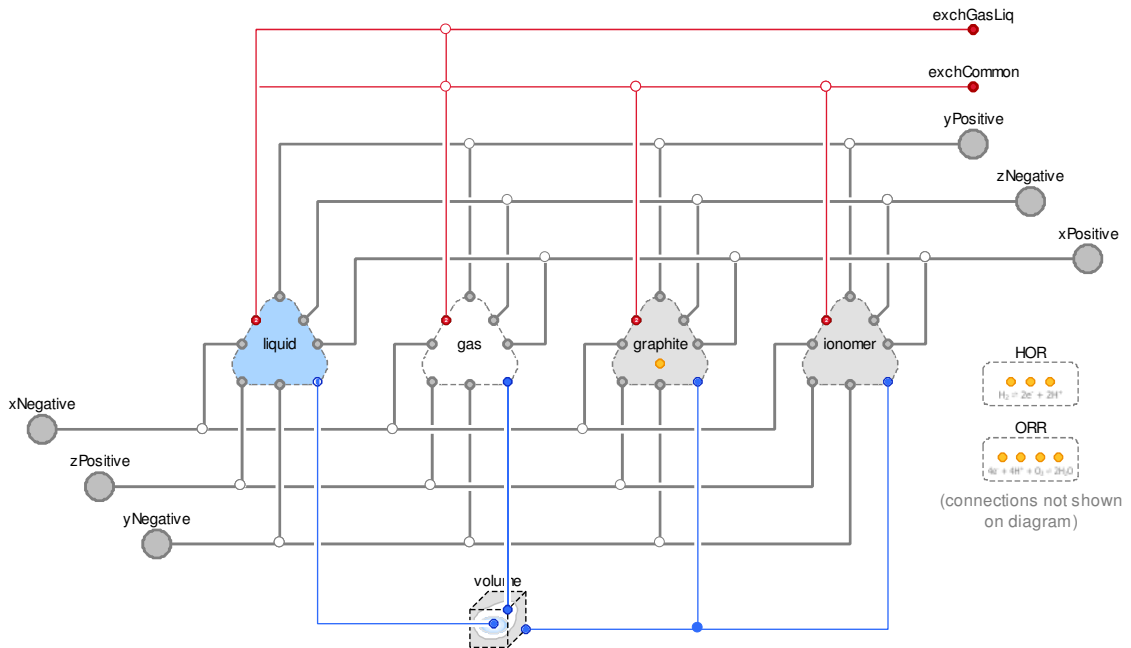
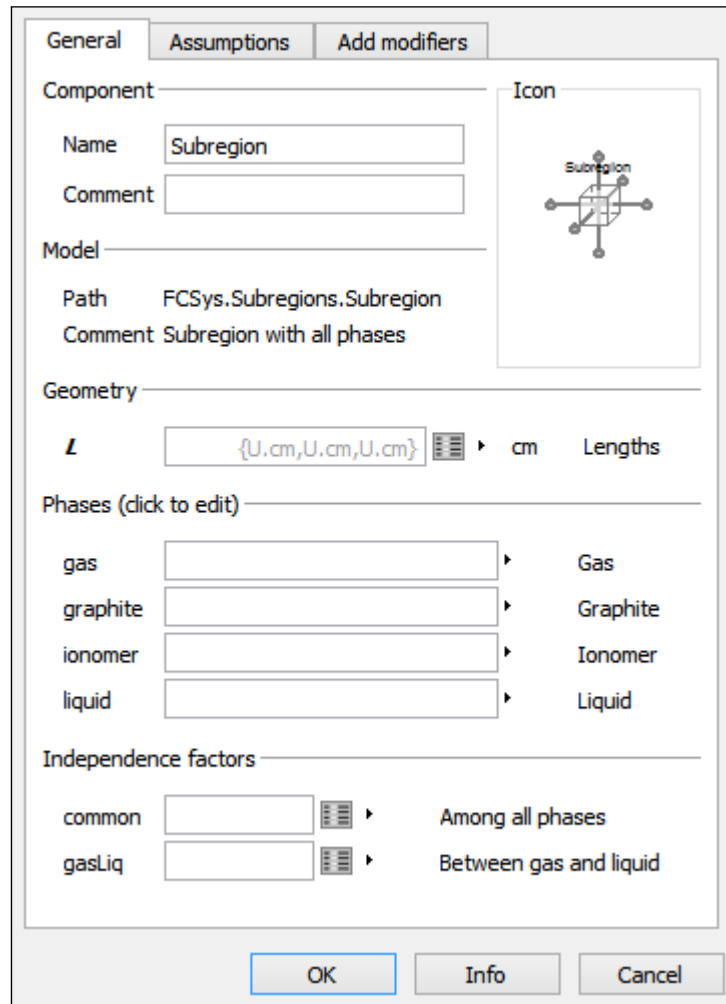


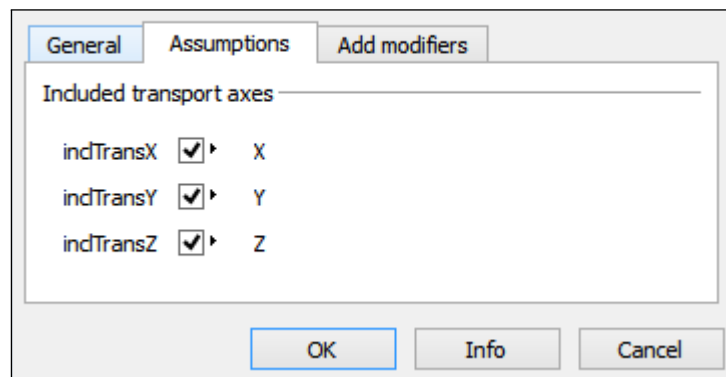
Figure 4.6: Diagram of a subregion.

The external connectors (`xNegative`, `xPositive`, etc.) represent the boundaries of the rectilinear control volume. Phases are labeled (`gas`, `liquid`, etc.) when they are connected

to the boundaries. By default, the Subregion model is three-dimensional (3D). However, assumptions may be applied via the Boolean parameters shown in Figure 4.7b (and listed in Table B.65) to individually eliminate components of translational momentum and pairs of boundary connectors. The settings for the phases (Figure 4.5) can be accessed through the main tab of the parameter dialog shown in Figure 4.7a.



(a) General parameters.



(b) Assumptions.

Figure 4.7: Parameter dialog for a subregion.

4.10 Regions/Layers

Related sections of the documentation:

- B.48–B.55 `FCSys.Regions.*`

The layers of the cell are represented by regions which contain 3D arrays of subregions. Each layer is an extension of a base `Region` model with the appropriate settings (e.g., geometry and selection of species). The grid is fixed and rectilinear but may have irregular spacing. Figure 4.8a shows the diagram of the `Region` model. The subregions icon in the center actually represents an array of subregions (which defaults to $1 \times 1 \times 1$). The interconnections are automatically included. Figure 4.8b shows the equivalent form for a $2 \times 2 \times 2$ array, without the external connectors. The boundaries (`xNegative`, `xPositive`, etc.) are two-dimensional (2D) arrays of bus connectors.

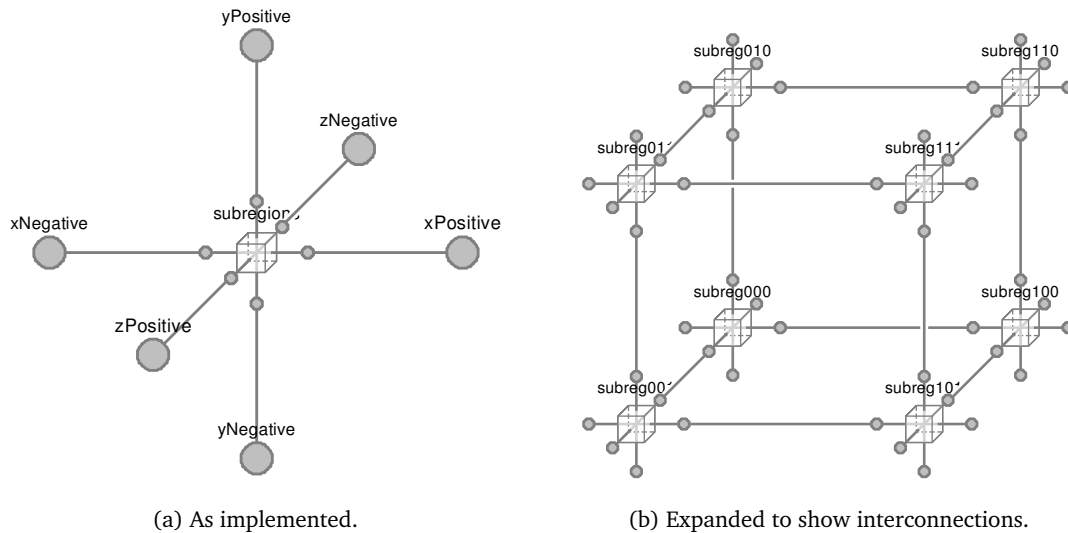
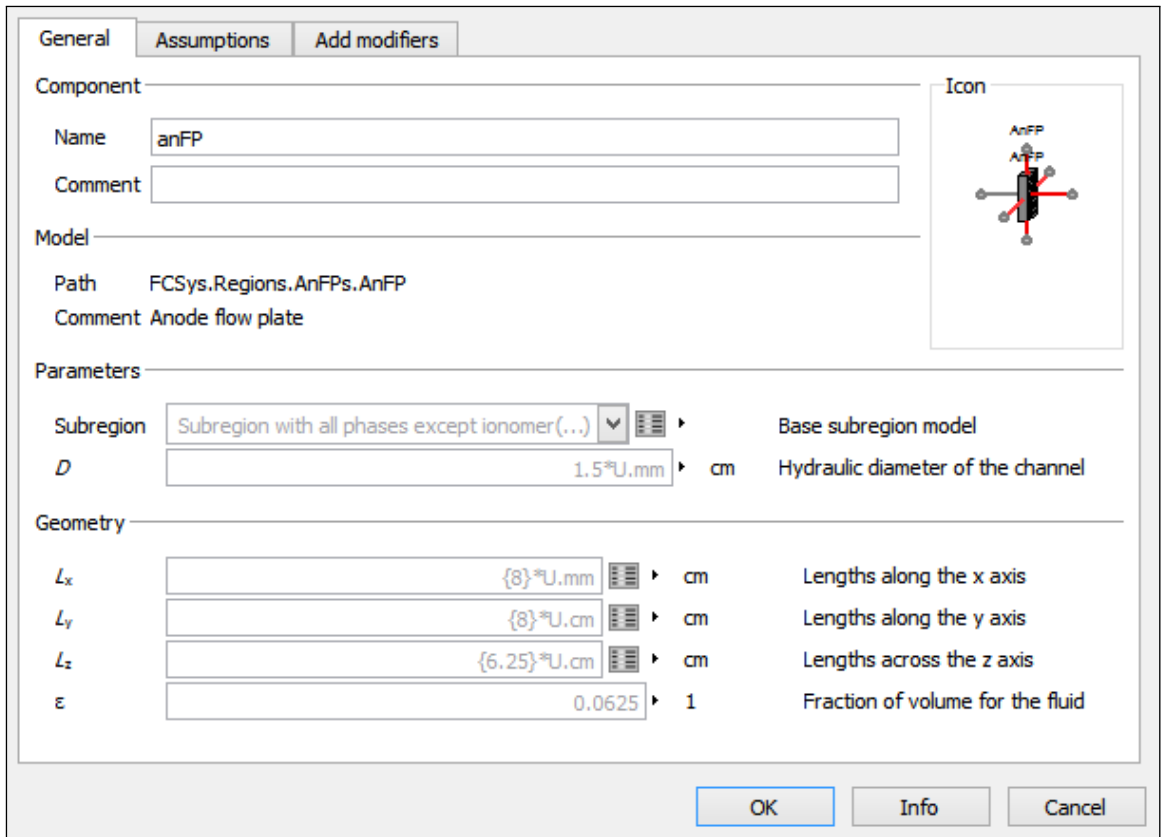
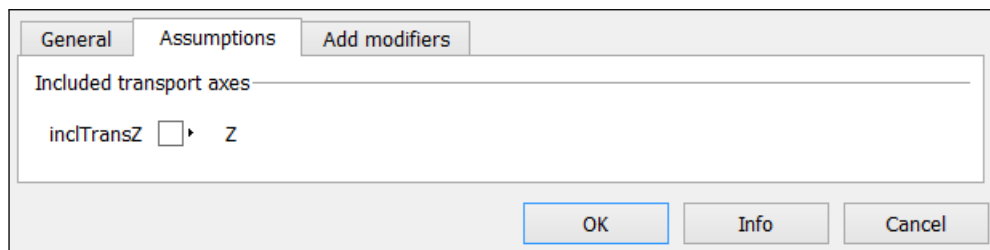


Figure 4.8: Diagrams of a region.

Figure 4.9 shows the parameter dialog for a region model—the anode flow plate. The lengths (L_y , L_x , and L_z) are one-dimensional arrays that contain the lengths of the subregions along the corresponding dimensions. Only the z-axis boundaries are optional (Figure 4.9b) because the x- and y-axis boundaries are required due to the cell geometry.



(a) General parameters.



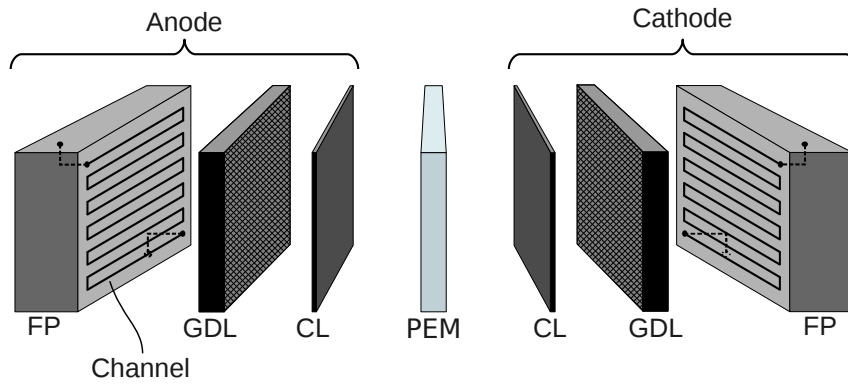
(b) Assumptions.

Figure 4.9: Parameter dialog for the anode flow plate.

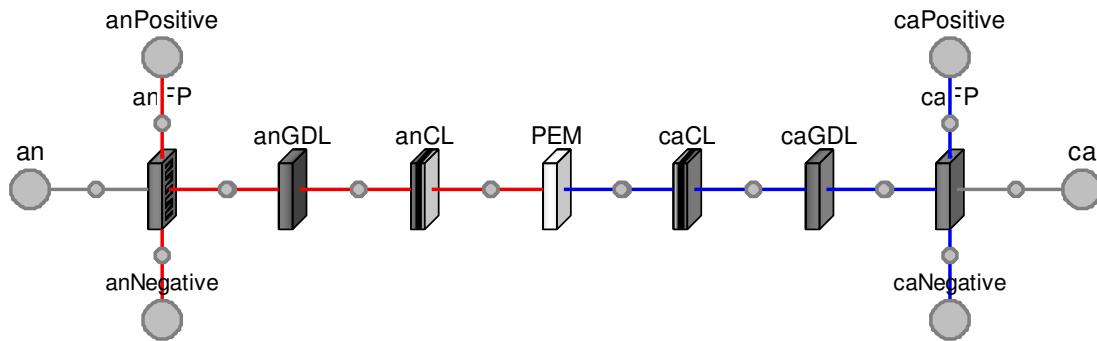
4.11 Assemblies/Cells

As shown in Figure 4.10, the diagram of the fuel cell model (Figure 4.10b) corresponds directly to the physical structure of the cell (Figure 4.10a and Figure 1.4). Figure 4.10c shows a version of the cell model with nearly minimal complexity given the present modeling framework. Its gas diffusion and catalyst layers are integrated (anCGDL and anCGDL) and all the species have the same temperature in every subregion, regardless of phase.

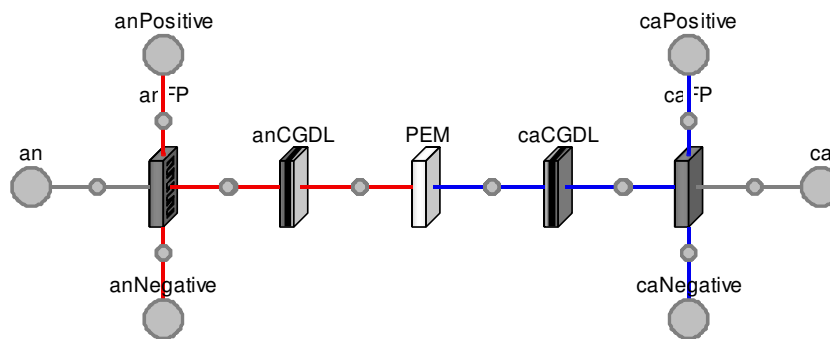
By default, each layer only contains one subregion (i.e., 1D model), but this can be increased. The number of sets of subregions and the lengths of the subregions in the y and z directions (along and across the channel) are the same for all of the layers. However, the number of sets of subregions and the lengths of the subregions are independent in the x direction (through the cell).



(a) Physical structure.



(b) Standard model diagram.



(c) Simplified model diagram.

Figure 4.10: Single-cell PEMFC.

4.12 Test Stand

Related section of the documentation:

- B.1–B.2 `FCSys.Assemblies.Cells.Examples.*`
- B.36 `FCSys.Conditions.Environment`

The highest physical level of the library is the `TestStand` model, which applies boundary conditions to the fuel cell. As shown in Figure 4.11, there are four boundary conditions for the channels—a source and a sink for both the anode and cathode. The cell model is bidirectional, so the choice of inlet and outlet can be switched via `anRouter` and `caRouter`. By default, the `anBC` and `caBC` models apply uniform temperature to the exterior of the flow plates along the x axis. An electrical model from the Modelica Standard Library [2] is connected as a load to the first (1, 1) segment of the cell in the yz plane via the `anAdapt` and `caAdapt` adapters. The default load is a current ramp (`Modelica.Electrical.Analog.Sources.CurrentRamp`).

Besides the electrical load, the following key conditions are adjustable:

1. Temperature of the end plates and reactant sources
2. Relative humidities of the reactant sources (independently adjustable)
3. Fixed or stoichiometrically-varying reactant flow rates (independently adjustable)
4. Fraction of N_2 in the cathode supply
5. Pressures of the reactant sinks
6. Optional inclusion of liquid water

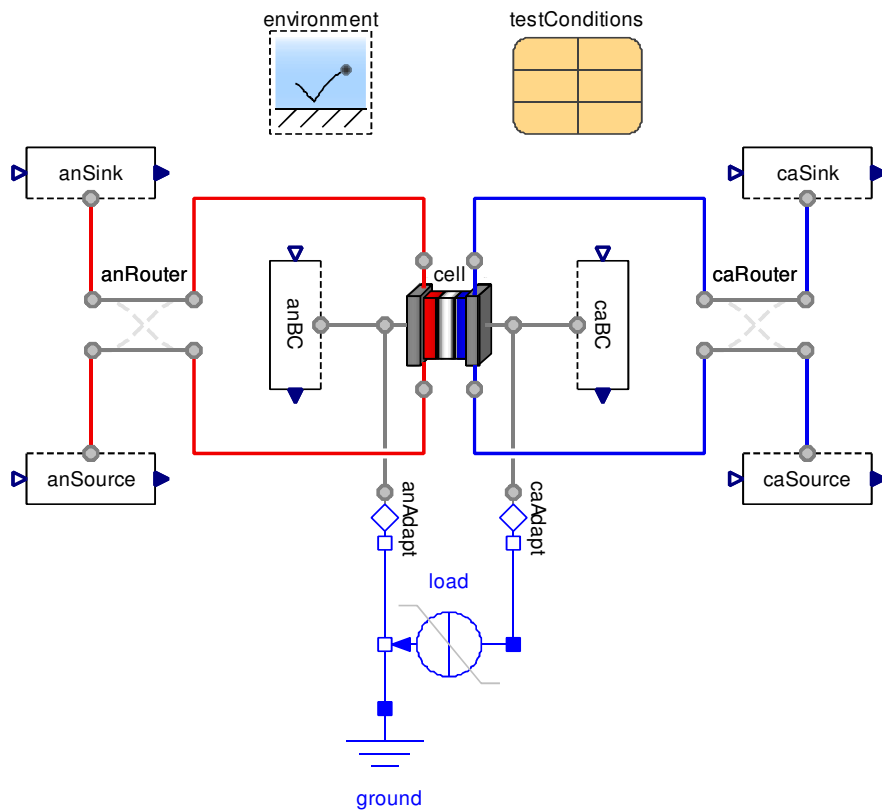


Figure 4.11: Diagram of the fuel cell test stand.

4.13 Summary

This chapter described the implementation of the model library in the Modelica language [1]. It provided references to the more detailed documentation in Appendix B. The model library, like this chapter, builds from low-level classes such as types and constants to high-level classes such as the fuel cell model. The models are highly modular and reconfigurable. In the next chapter (Chapter 5), we will provide basic examples before exercising the full fuel cell model in Chapter 6.

BASIC EXAMPLES

The purpose of this chapter is to provide initial, low-level validation of the model library. It presents some basic examples of the model introduced in Chapters 3 and 4. Each section begins with an introduction of the physical conditions and assumptions. Then, simulation results are shown and discussed. Most of the examples in this chapter are not intended to be representative of a fuel cell. However, the last two examples demonstrate phase change processes which are scaled so that the time constants roughly correspond to a proton exchange membrane fuel cell (PEMFC).

It is significant that all of the following examples—from gas dynamics to electrical conduction—were generated from the same working model (Subregion), only instantiated with various geometries and initial and boundary conditions. The models of solids, liquid water, gases, electrons, and protons are different only in their material properties and default assumptions, all of which are configurable.

In addition to these examples, a comprehensive package of test models was created and simulated to verify the thermodynamic and transport properties of the fluids against [150] and [172]. Also, the physical constants were verified against [188].

The results of each example begins with statistics about the model and its simulation. The models contain a number of variables, some of which are time-varying. The number of states is the number of time derivatives that are algebraically independent. It is the number of unique ways that energy can be introduced and stored in a model (enthalpy of formation between phases, boundary work, energy due to charge displacement, kinetic energy along each axis, and internal energy). At a given time, the state variables can be considered known. The size of each system of equations is the number of algebraically coupled equations that must be solved from the known variables, which include the states and time-constant variables. The systems of equations may be linear or nonlinear.

Before simulation, a model must be translated. This includes the process of parsing the Modelica syntax, flattening the object-oriented code, manipulating and sorting the equations, tearing systems of equations, and compiling the result with links to the solvers and integrator. The models were translated and simulated on a Samsung ATIV Book 8 laptop computer (Intel Core i7 3635QM @ 2.40 GHz, 8 GB RAM DDR3 @ 1600 MHz, 5400 RPM hard drive) running Windows 8.1 and Dymola 2014. The hardware represents technology that is approximately one year old (processor introduced 3rd quarter 2012).

The default Dymola settings were used, including the Differential/Algebraic System Solver Library (DASSL) [195] and 500 output intervals, except:

1. GuardedSqrt = `false`
2. ImprovedPackageConstants = `true`
3. IncludeLibrariesForSimulink = `false`
4. SubstituteVariablesUsedOnce = `true`

Most models were simulated with an integration tolerance of 10^{-4} , but this was increased as needed. The following debugging options were enabled which may have affected the translation and compilation time:

1. CheckUnits = `false`
2. LogDefaultInitialConditions = `true`
3. LogStateSelection = `true`
4. OutputCPUtime = `true`
5. OutputModelicaCode = `true`
6. OutputModelicaCodeWithAliasVariables = `true`
7. StoreProtectedVariables = `true`

5.1 *Internal Flow*

This example compares the model to Poiseuille's law [177] for pressure drop along a pipe under laminar flow. Poiseuille's law was derived from the model equations in Section 3.7.3, so this example is partly a test that the implementation performs as expected. It also demonstrates inertance and thermal dynamics.

5.1.1 Conditions

Liquid water is forced along the long axis through a rectangular $1\text{ m} \times 1\text{ mm} \times 1\text{ mm}$ subregion as shown in Figure 5.1. In addition to a large-signal volumetric flow rate of $1.5\text{ cm}^3/\text{s}$, there is a small sinusoidal signal with an amplitude of $0.3\text{ cm}^3/\text{s}$ and a frequency of 1 Hz . This corresponds to an area-average (plug flow equivalent) velocity of $1.5 \pm 0.3\text{ m/s}$ as shown in Figure 5.2. A no-slip (zero velocity) condition is applied around the perimeter. The fluid source is held at $25\text{ }^\circ\text{C}$. There is no thermal conduction across the outlet (only advection) or around the perimeter.

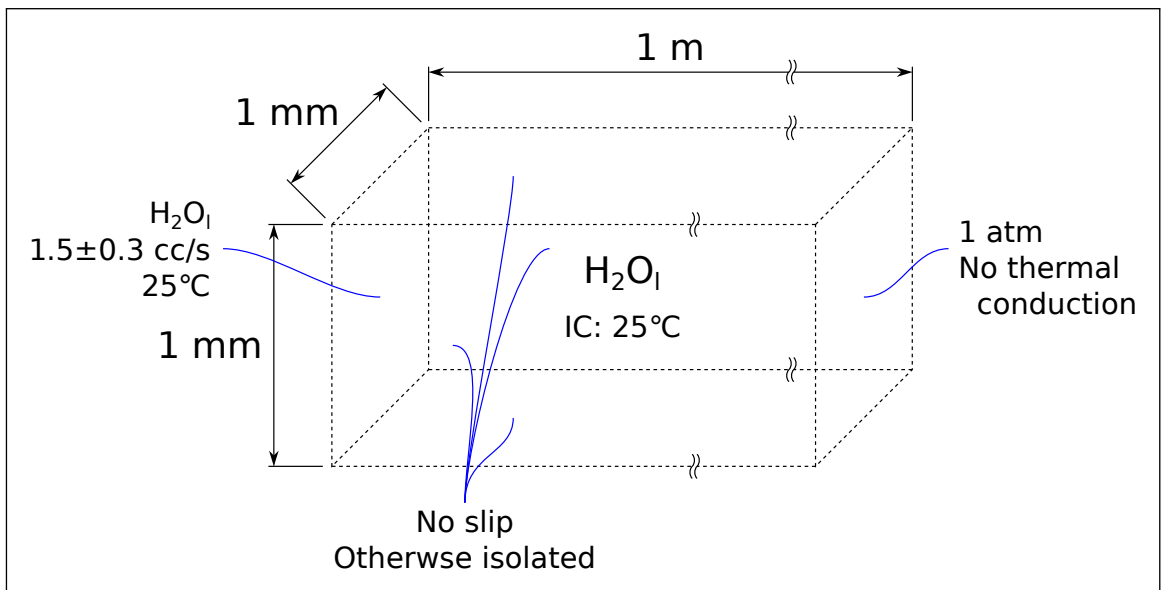
The Reynolds number varies from 1400 to 2100; therefore, laminar flow is expected. The translational Nusselt number for the liquid water is set to four ($Nu_\phi = 4$, the default), which is appropriate for laminar flow with a single subregion over the cross section.

5.1.2 Results and Discussion

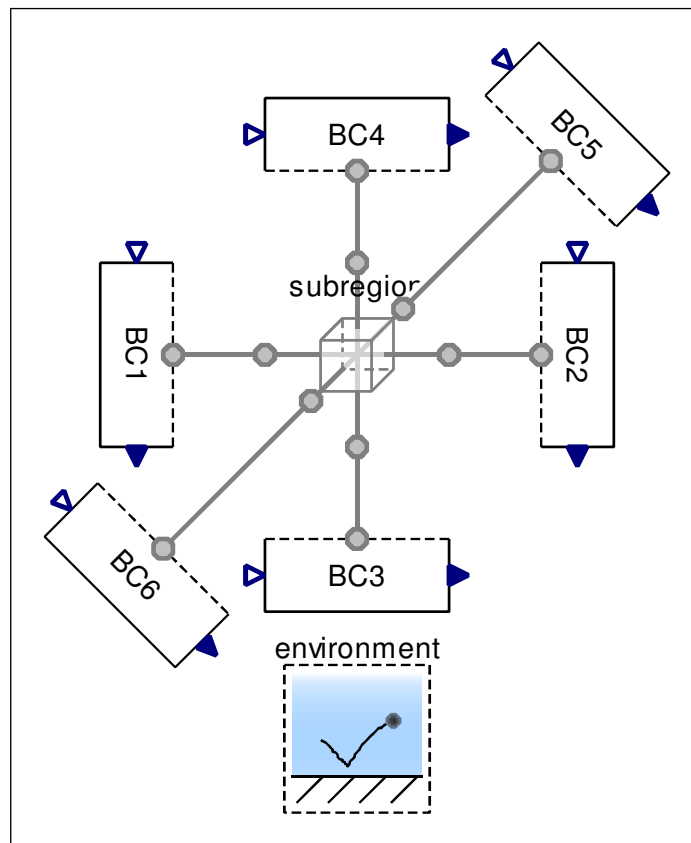
Table 5.1: Modeling and simulation statistics for the internal flow example.

	With analysis	Without analysis
Number of variables	516	474
Number of time-varying variables	104	55
Number of states	1	1
Sizes of the nonlinear systems of equations	None	None
Sizes of the linear systems of equations	4 sets of 2	4 sets of 2
Translation time	3 s	1 s
Simulation time	0.007 s	0.007 s

The modeling and simulation statistics are listed above. The model contains a fairly large number of variables (516) given the simple physical system. The object-oriented nature of the model introduces overhead in terms of the number of variables. Some of the variables (8% of all variables and 47% of the time-varying variables) are outputs that are purely for analysis. They can be disabled without affecting the behavior. Roughly one fifth of the variables are time-varying. The model contains only one state: temperature. Velocity is not a state since the translational dynamics are imposed directly by the boundary conditions (prescribed volumetric flow rate) and the fluid is incompressible. There are no time-varying systems of algebraic equations after manipulation. The model simulates very quickly (7 ms). The analysis variables



(a) Physical domain.



(b) Model.

Figure 5.1: Configuration of the internal flow example.

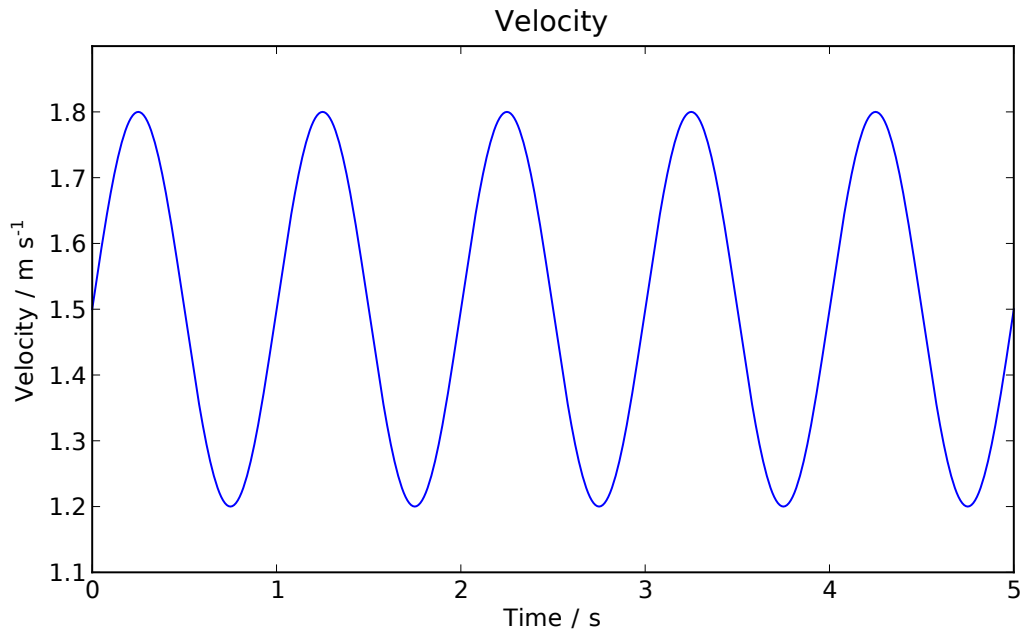


Figure 5.2: Fluid velocity for the internal flow example.

have negligible effect on the simulation time. Translation takes over 400 times longer than simulation. However, the model may be re-simulated with various parameter settings without re-translating it.

Figure 5.3 shows the difference in thermodynamic pressure across the subregion. Due to inertance, the pressure difference of the model leads the pressure difference according to Poiseuille’s law and has a slightly larger amplitude. The effect of inertance becomes more prominent at higher frequencies and less so at lower frequencies.

The temperature of the fluid in the subregion increases due to viscous dissipation as shown in Figure 5.4. However, the temperature rise is only 9 mK (to 25.009 °C) because the rate of heat generation is much smaller than the rate of enthalpy flow from the source. The increase is so slight that the simulation tolerance must be increased from 10^{-4} (the default) to 10^{-8} in order to distinguish it from the solver error. The temperature increases with a first-order dynamic response. A sinusoidal temperature variation is superimposed due to the small-signal part of the flow rate.

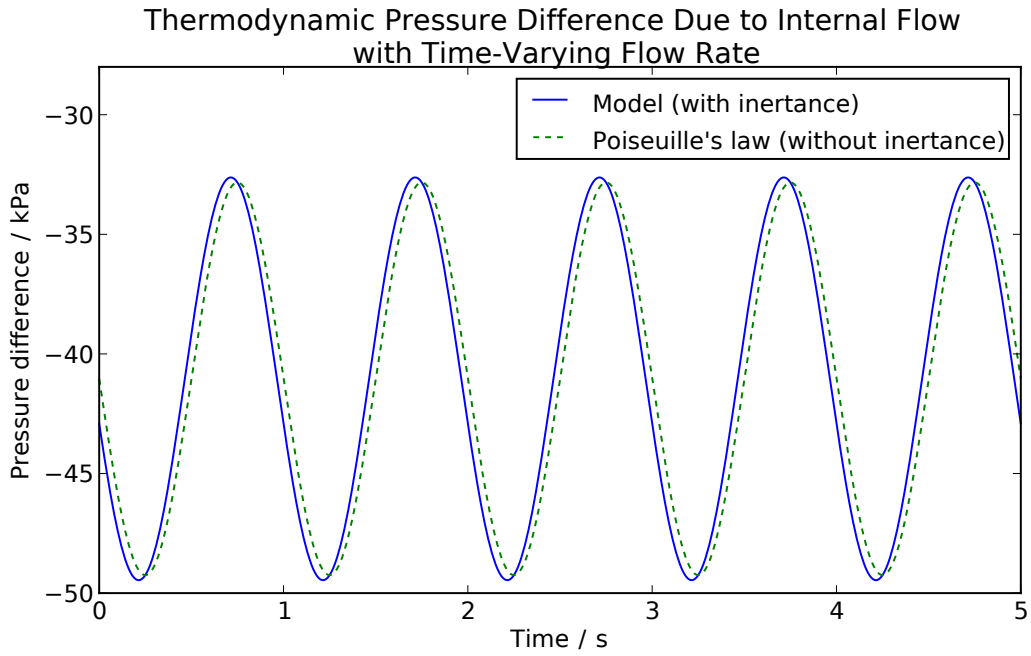


Figure 5.3: Dynamic pressure difference under internal flow.

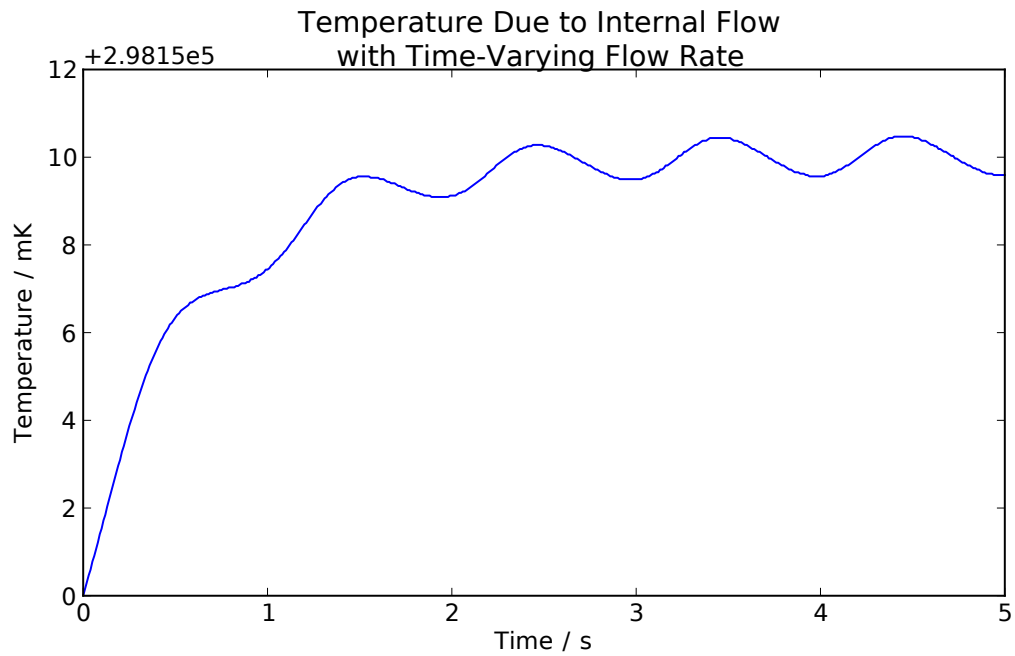


Figure 5.4: Thermal transients due to viscous dissipation in internal flow.

5.2 Echo

This example shows the dynamics of compressible flow in the model. A pressure wave reflects between the external boundaries of adjacent subregions. The effect of the discretization scheme can be seen. The key model equations are material transport (Equation 3.104) and material conservation (Equation 3.25).

5.2.1 Conditions

Two cubic 1 cm^3 subregions containing hydrogen (H_2) are arranged side-by-side with an initial pressure difference of 100 Pa as shown in Figure 5.5. The `subregions` array of models is removed and bypassed upon translation because there are only two subregions. Both subregions are initialized to 25°C . All external boundaries are isolated (i.e., closed, adiabatic, and free-slip). The model is evaluated with upstream discretization and with the central difference scheme.

5.2.2 Results and Discussion

Table 5.2: Modeling and simulation statistics for the echo example.

	Upstream discretization	Central difference
Number of variables	488	488
Number of time-varying variables	122	118
Number of states	5	5
Sizes of the nonlinear systems of equations	None	None
Sizes of the linear systems of equations	2 sets of 2	2 sets of 2
Translation time	3 s	3 s
Simulation time	0.011 s	0.010 s

The model simulates slightly faster (9%) with the central difference scheme, although the time may not be significant given the uncertainty due to the overhead of the operating system. These models are roughly the same as the internal flow model (Section 5.1) in terms of complexity, translation time, and simulation time.

As shown in Figure 5.6a, the model exhibits oscillations due to the coupled dynamics of compression and translation. This is essentially a mass-spring-damper system. The damping is

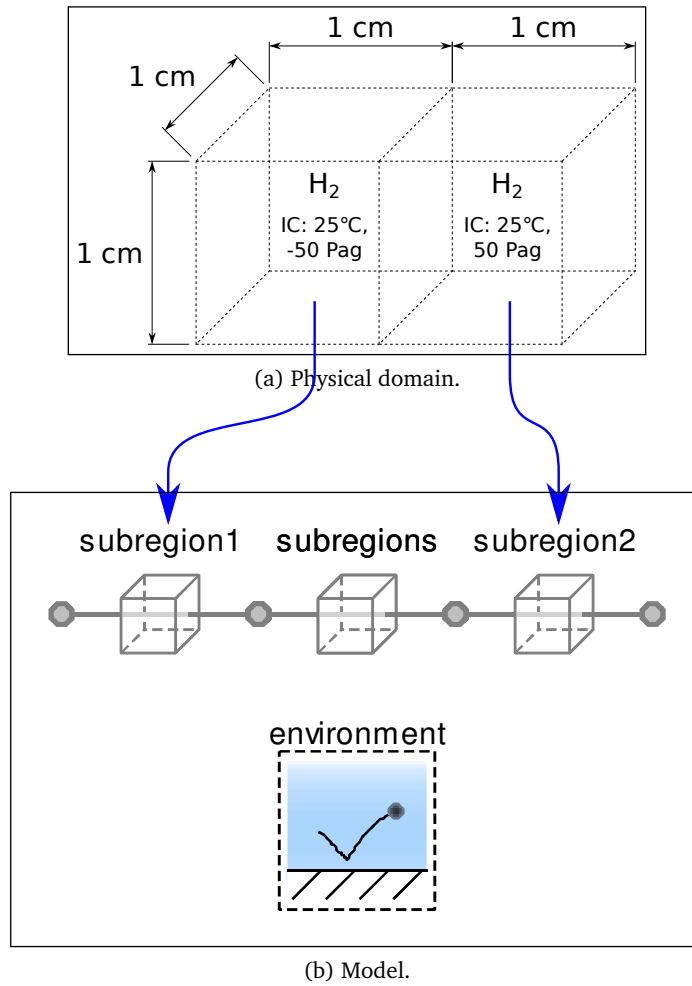
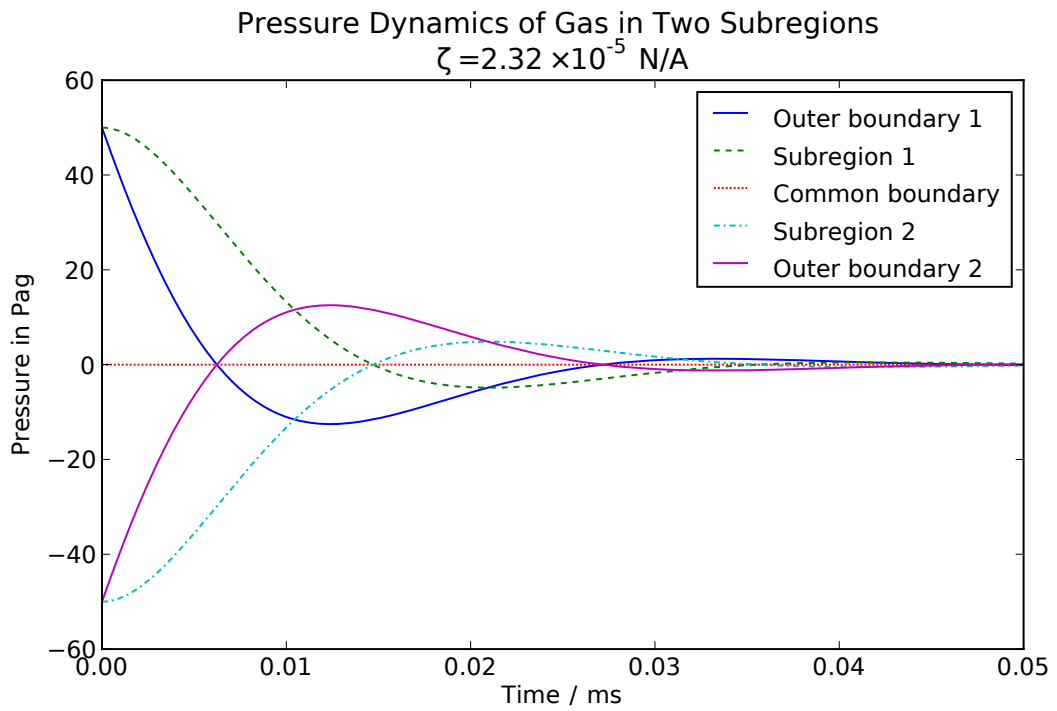


Figure 5.5: Configuration of the echo example (H_2 gas with an initial pressure difference).

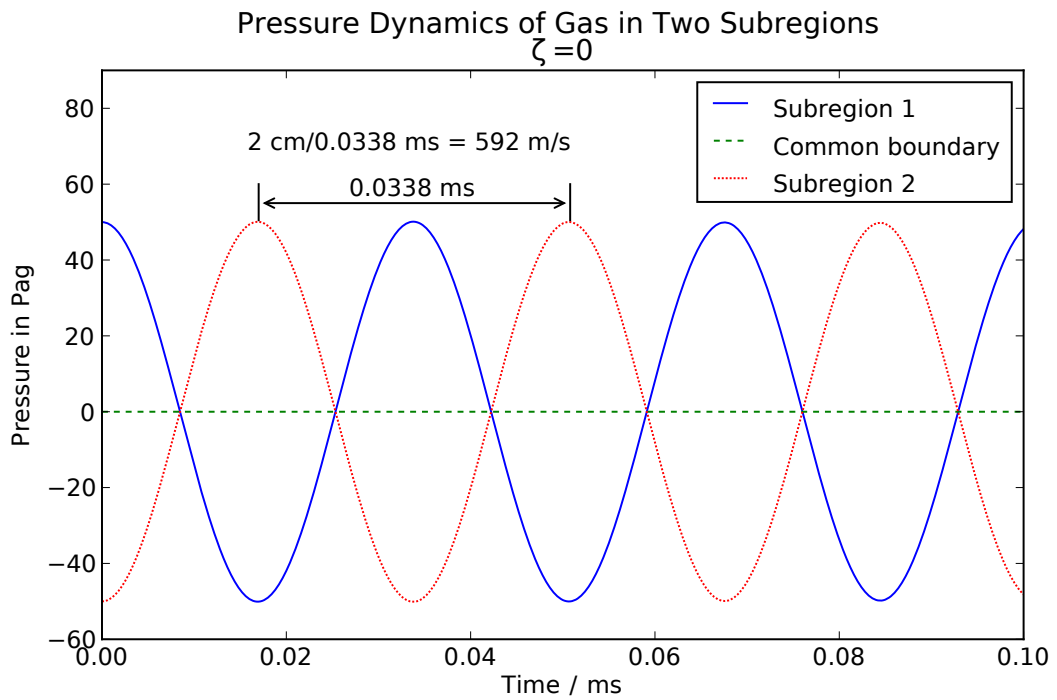
due to irreversible material compression and expansion (Equation 3.104). If the continuity (ζ) is set to zero, there is no damping. This case is shown in Figure 5.6b. The outer boundaries have the same temperature as the nearest subregion. The period of oscillation is 0.0338 ms, which corresponds to a wave velocity of 592 m/s. This is roughly half of the speed that is expected for H_2 as an ideal gas at 25 °C with an adiabatic index of approximately 1.4. Currently, the reason for the discrepancy is unknown. The plots in Figure 5.6 are from the case where upstream discretization is used, but the central difference scheme gives identical results. The common boundary remains at the average pressure in both cases. An upstream bias is applied to the balance between pressure gradients and irreversible compression, but the associated Péclet number is very small even with the significant damping evident in Figure 5.6a. For incompressible fluids, the damping and the Péclet number is exactly zero.

Due to thermodynamics, the temperature of the first subregion decreases as its gas expands and the temperature of the second subregion increases as it is compressed. The changes in temperature are small because the thermal effect is secondary to the main effect of pressure equalization. As shown in Figure 5.7, the temperature at the common boundary depends on the discretization scheme. According to upstream discretization, the temperature of the common boundary is biased by the source (Figure 5.7a). There is no such bias with the central difference scheme (Figure 5.7b). Both of these plots are with the default value of continuity ($\zeta = 2.32 \times 10^{-5} \text{ N/A}$). Since the outer boundaries are adiabatic, the temperature at the outer boundaries are equal to the temperature of the nearest subregion. At the end of these plots (0.05 ms), the temperatures of the regions are different, yet there is very little thermal convection because the velocities have decayed to nearly zero. Over a much longer period (2.5 s), the temperatures equalize due to thermal conduction, as shown in Figure 5.8. The long-term trend of temperature is the same for upstream discretization and the central difference scheme.

Gas travels in the positive direction due to the initial pressure difference and then returns due to compression, as shown in Figure 5.9. Although it is not apparent in Figure 5.9a, the magnitude of the velocity is slightly greater in the second subregion because that subregion initially has less mass and therefore greater acceleration from a given pressure difference. This

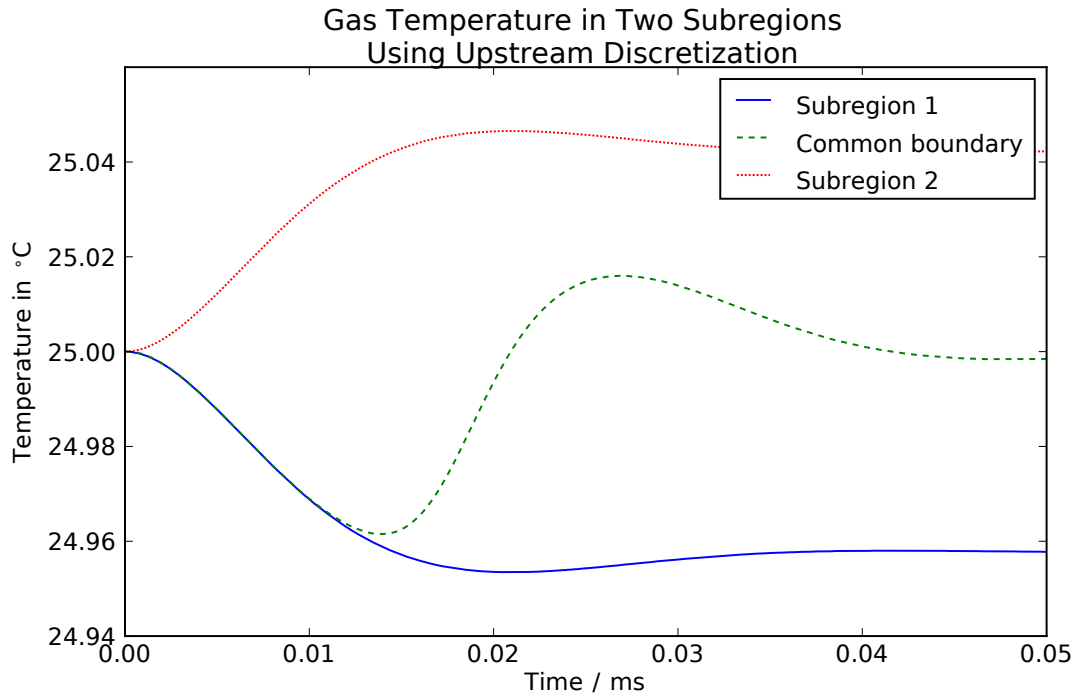


(a) Default damping.

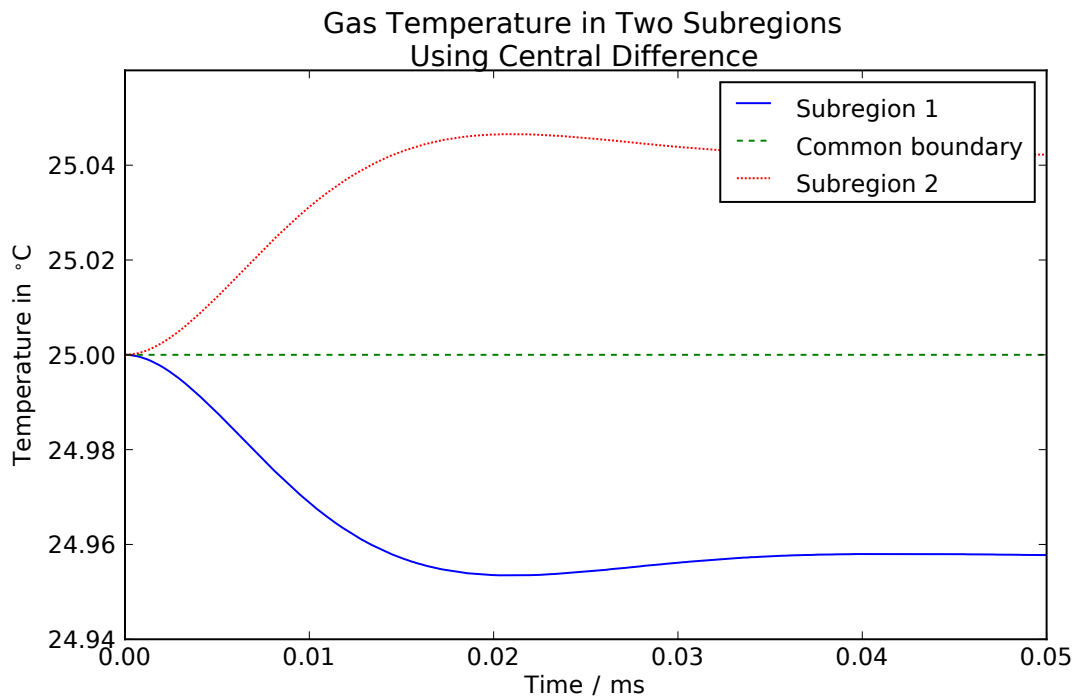


(b) No damping.

Figure 5.6: Pressure dynamics in the echo example.



(a) Upstream discretization.



(b) Central difference scheme.

Figure 5.7: Temperature oscillations in the echo example.

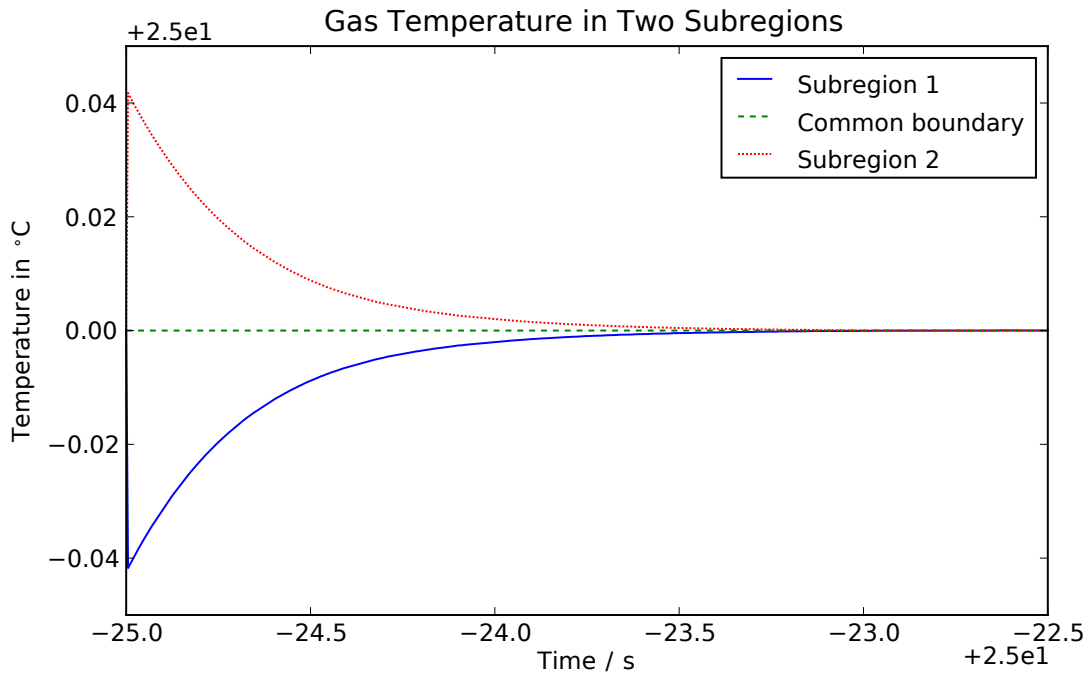
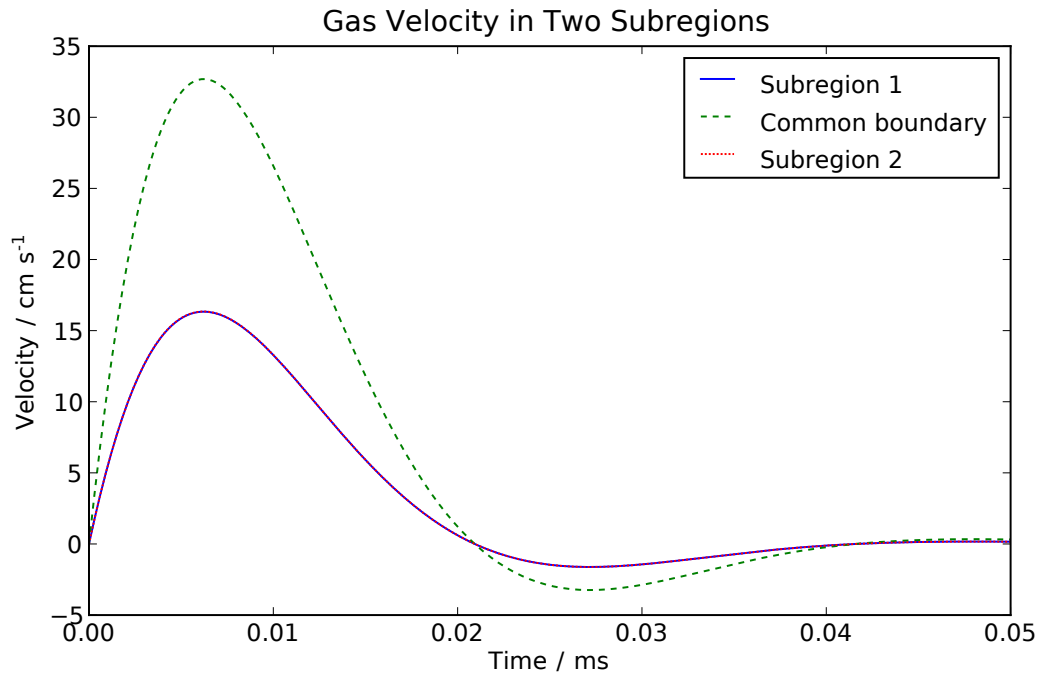
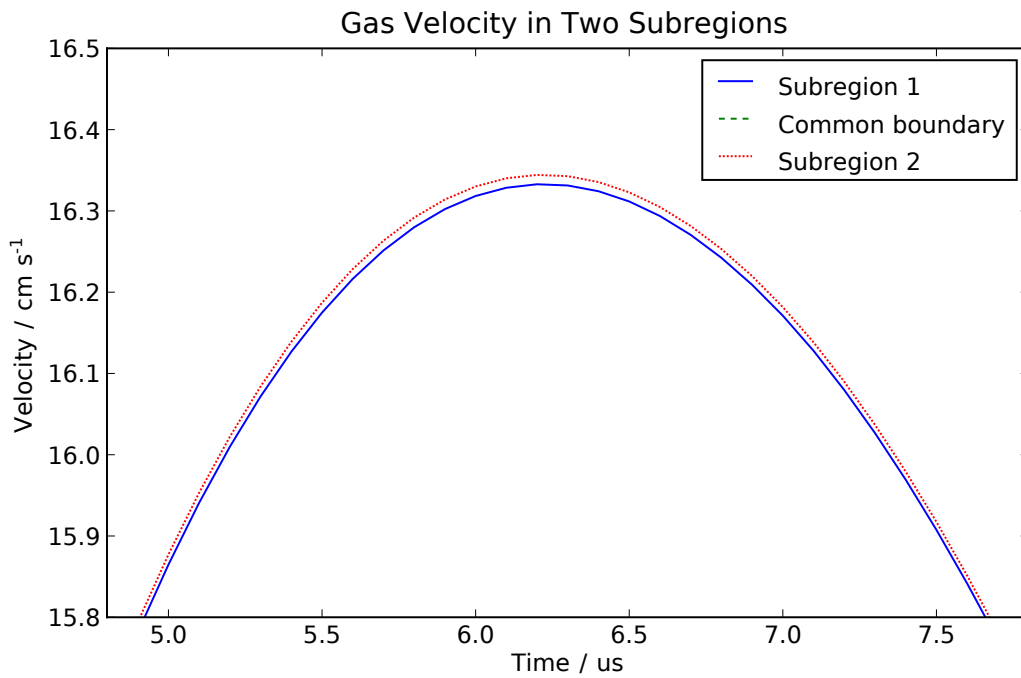


Figure 5.8: Long-term thermal equilibration in the echo example.

is shown in Figure 5.9b. The plots are for the case of upstream discretization, but the central difference scheme gives identical results.



(a) Dampened oscillation.



(b) Close-up of first crest.

Figure 5.9: Velocity in the echo example.

5.3 Air Column

This example shows the effect of body forces on a gas. Like the previous example, it exhibits oscillations due to the coupled dynamics of translation and compression. The key model equations are material transport (Equation 3.104), material conservation (Equation 3.25), and the translational momentum balance (Equation 3.28).

5.3.1 Conditions

A vertical column of five $10\text{ m} \times 10\text{ m} \times 10\text{ m}$ subregions contain nitrogen (N_2) gas (roughly representing air), as shown in Figure 5.10. The gas is initialized uniformly to 25°C and 1 bar. The external boundaries are isolated (i.e., closed, adiabatic, and free-slip) except for the upper boundary which is held at 25°C and 1 bar. Standard gravity is applied (9.81 m/s^2). The central difference scheme is used.

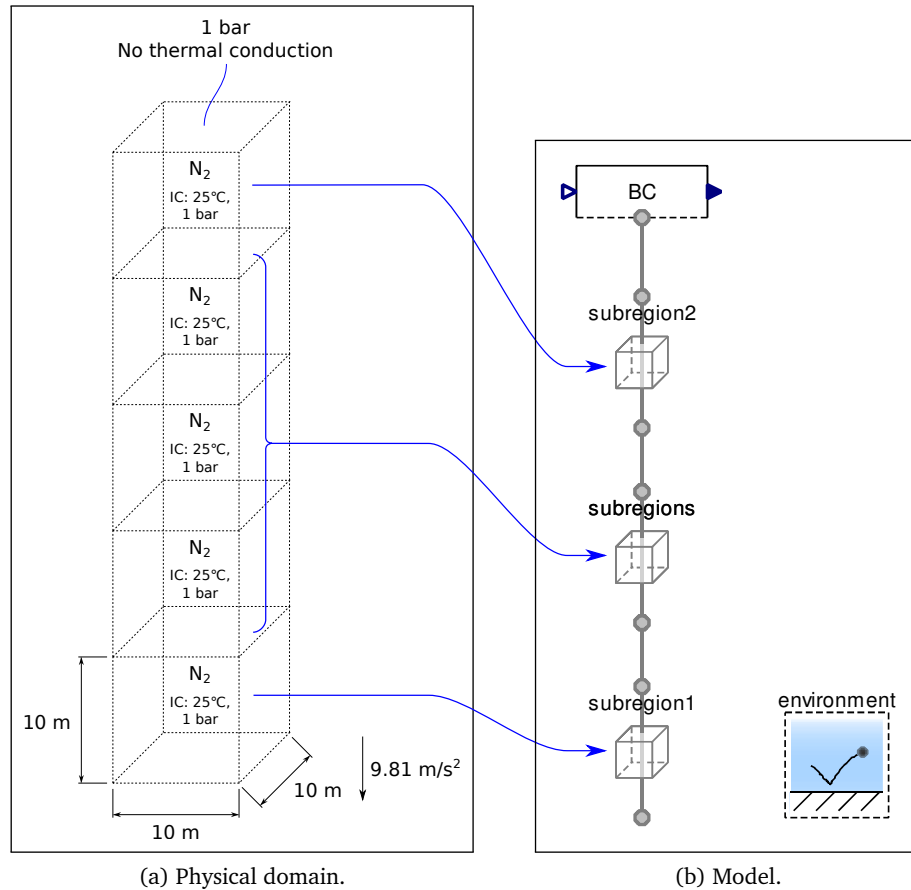


Figure 5.10: Configuration of the air column example (gas under the influence of gravity).

5.3.2 Results and Discussion

Modeling and simulation statistics:

- Number of variables: 1168
- Number of time-varying variables: 284
- Number of states: 15
- Sizes of the nonlinear systems of equations: None
- Sizes of the linear systems of equations: 8 sets of 2
- Translation time: 3 s
- Simulation time: 0.016 s

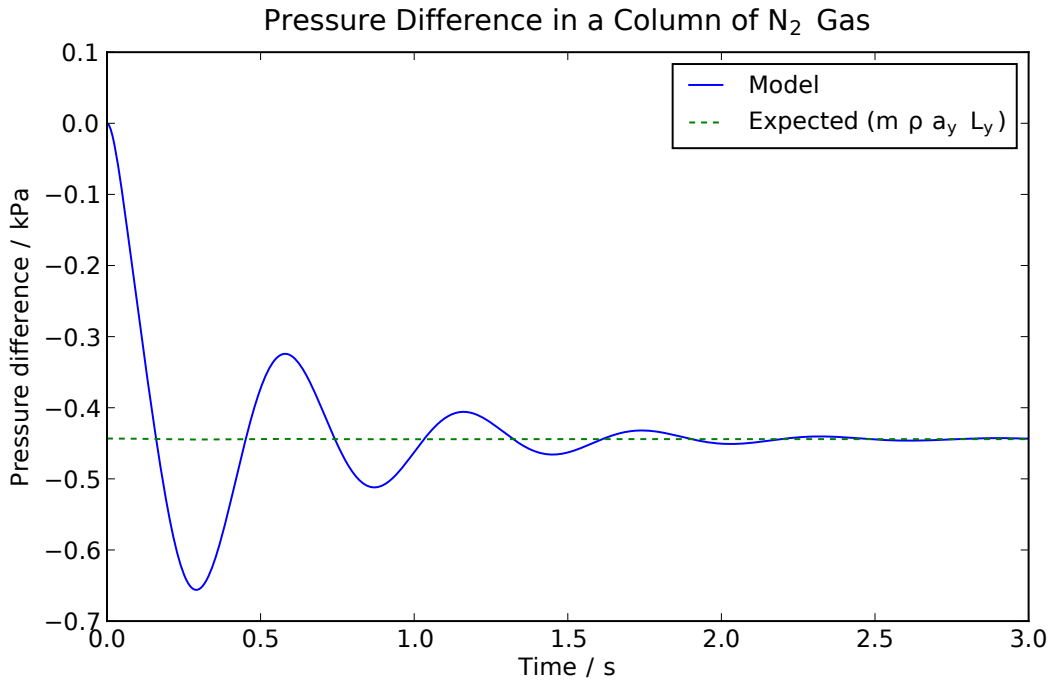
This model has roughly twice as many variables as the previous examples, yet it translates and simulates in about the same time. There are 15 states—the temperature, pressure, and vertical component of velocity in each of the five subregions.

The total pressure difference over all the subregions is plotted in Figure 5.11a. It oscillates due to the dynamics of translation and compression. It takes approximately 3 s for the pressure difference to settle to the expected value—the product of density, acceleration due to gravity, and column height ($m\rho a_y L_y$). Figure 5.11b shows that the pressure gradient is uniform at steady state, as expected.

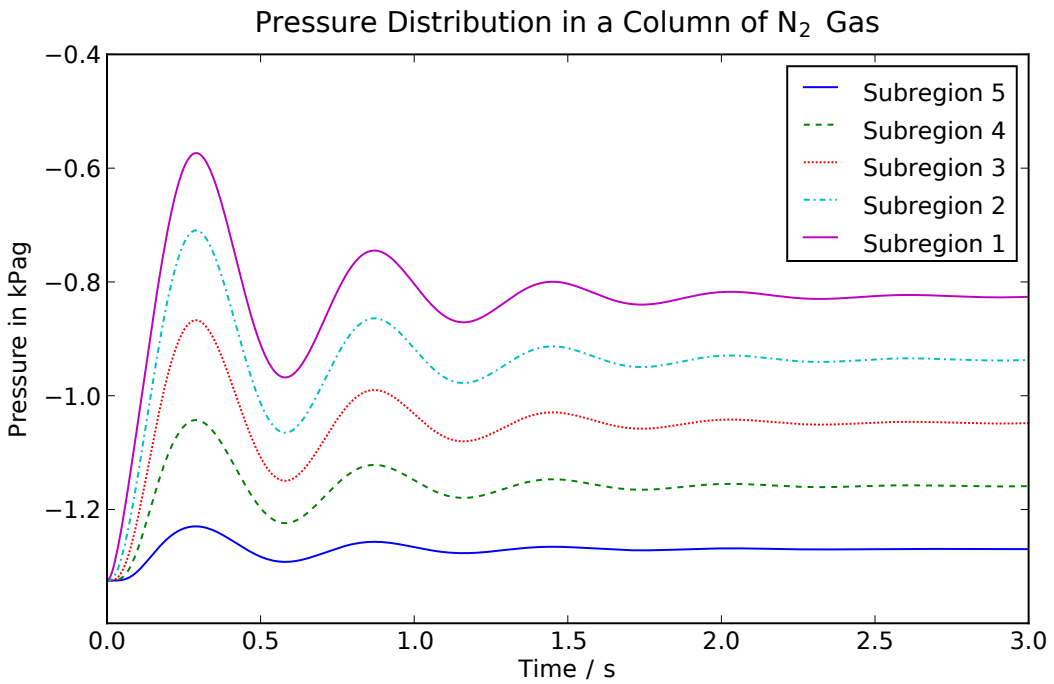
As shown in Figure 5.12, the gas accelerates downward and then rebounds upward due to compression. The maximum downward velocity is approximately 85 cm/s at the upper boundary.

The temperatures of the subregions increase due to the adiabatic compression, as shown in Figure 5.13a. This develops a temperature gradient where the lower subregions are slightly hotter. In reality, this would lead to natural convection. However, in this idealized model, there is only one transport path and no way for a circulation cell to develop. Therefore, the temperature can only equalize by conduction.ⁱ The time constants, which are calculated as output variables of the model and plotted in Figure 5.14, are approximately 19 days for transport from

ⁱThe model also assumes no thermal radiation.



(a) Total pressure difference.



(b) Pressure distribution.

Figure 5.11: Pressure in the air column example.

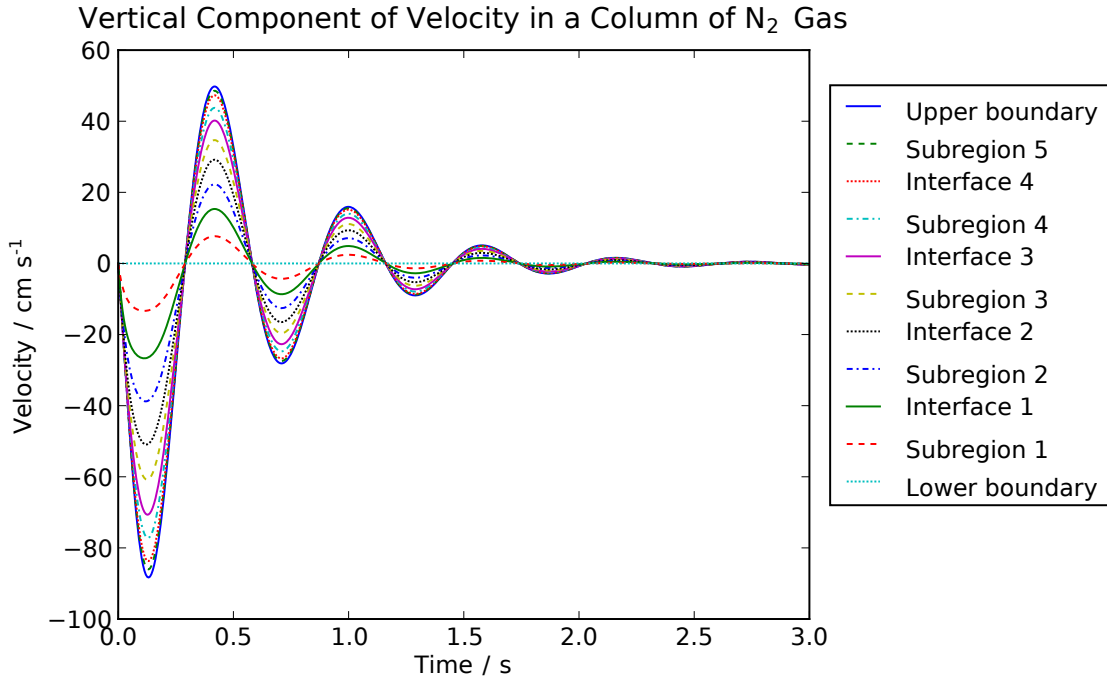
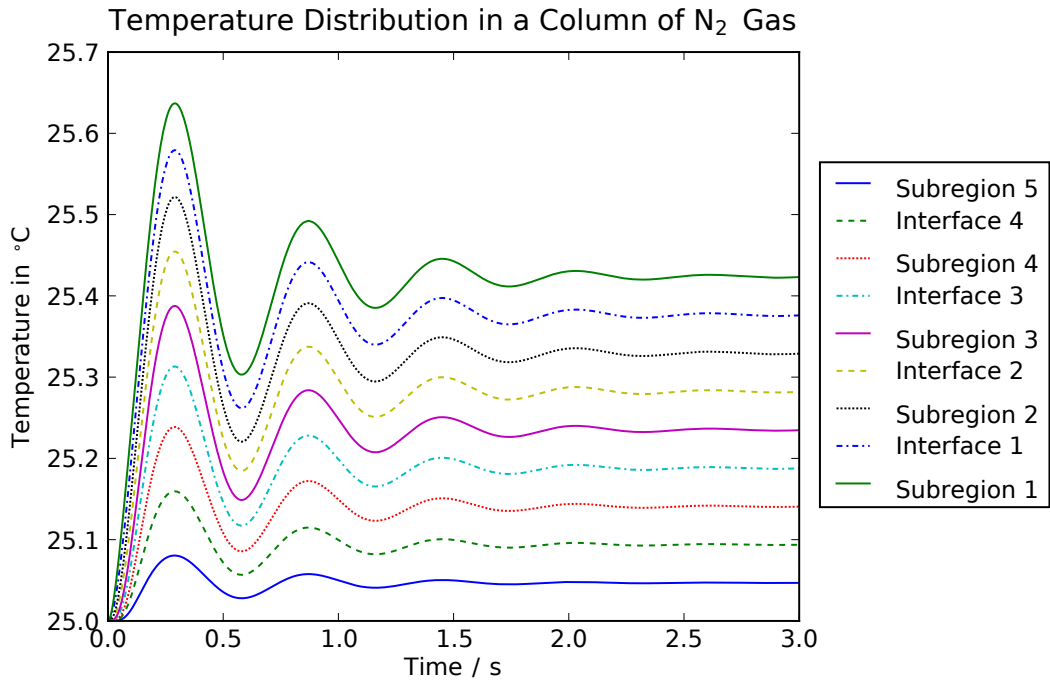
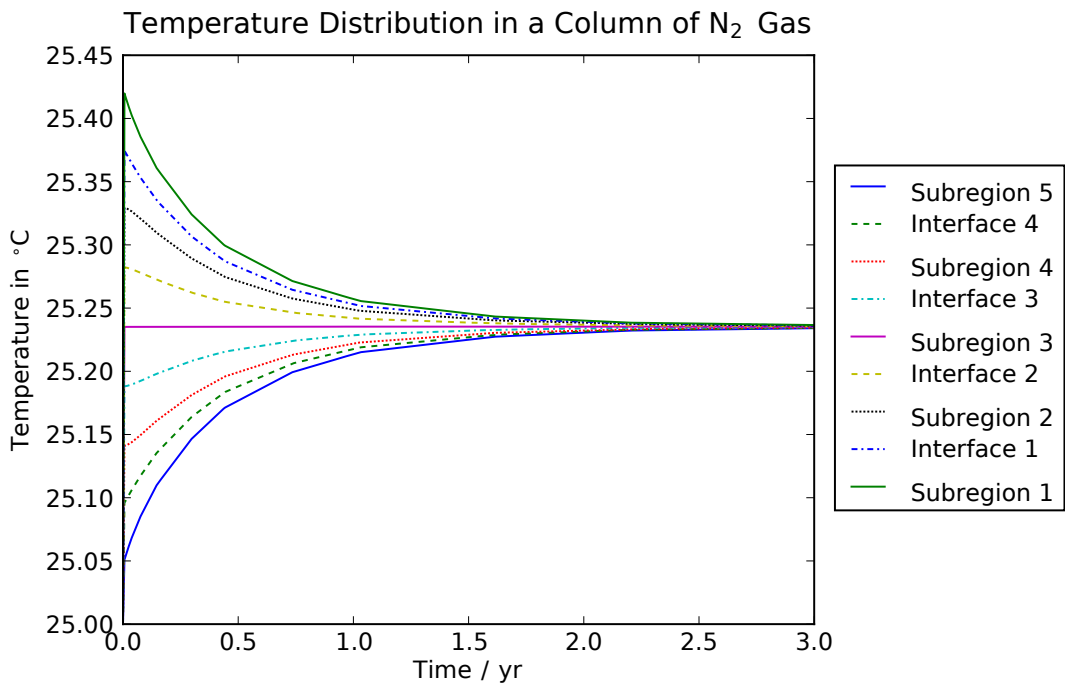


Figure 5.12: Velocity transients in the air column example.

a boundary into a subregion due to the size of the system and the properties of the gas. These large time constants do in fact match the theoretical result. Figure 5.13b shows the long-term temperature trends. The full equalization takes approximately three years. While this example is rather academic, it demonstrates the stability and flexibility of the model and the solver. The same translated model was used for both the short- and long-term simulations—a ratio of 31,557,600 in simulated time—with roughly the same computation time (16 and 19 ms).



(a) Initial oscillations.



(b) Equalization over the long term.

Figure 5.13: Thermal transients in the air column example. Since the boundaries are adiabatic, the temperatures of the lower and upper boundaries (not plotted) are equal to the temperatures of subregions 1 and 5.

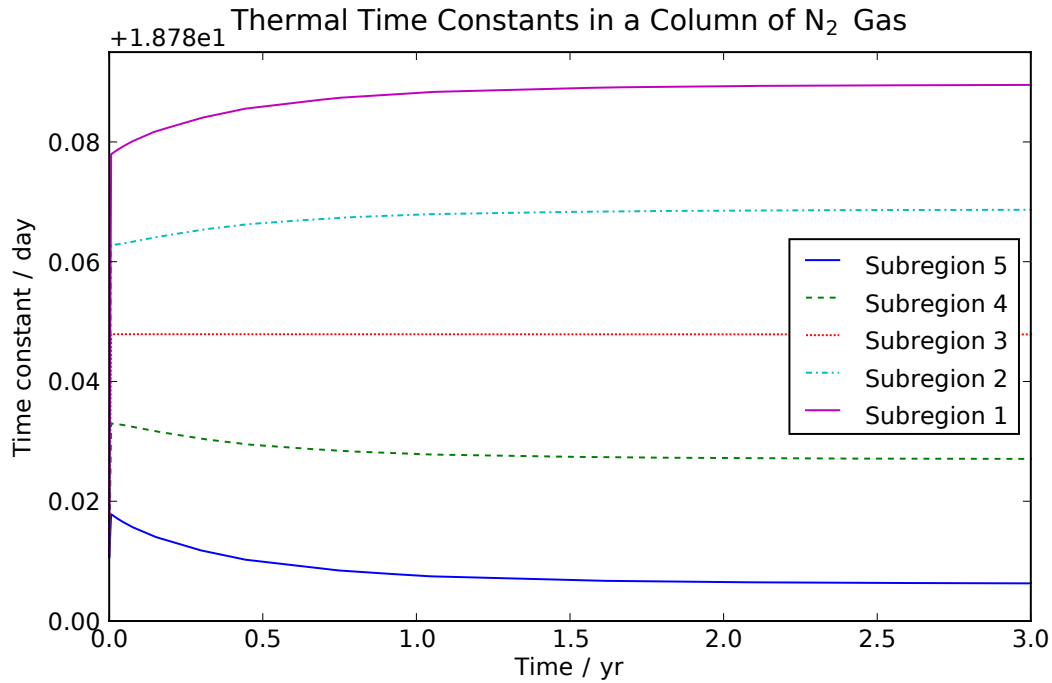


Figure 5.14: Thermal time constants in the air column example.

5.4 Electrical Conduction

This is an example of Ohm's law with thermal dynamics. The key equations are translational exchange (Equation 3.55), thermal conduction (Equation 3.145), and the energy balance (Equation 3.200).

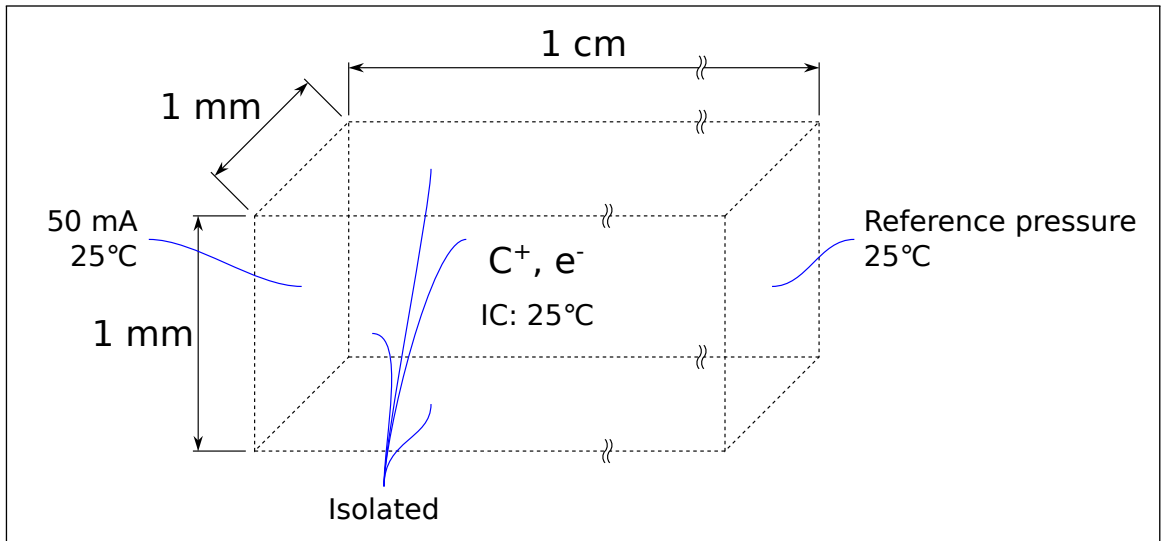
5.4.1 Conditions

A $1\text{ m} \times 1\text{ mm} \times 1\text{ mm}$ block of graphite is held at 25°C , as shown in Figure 5.15. An electrical current (zI) of 50 mA is pulled from the negative boundary (e^- into the boundary), and the positive boundary is held at a reference electronic pressure/potential. The electrical conductivity (σ) is 100 S/m . The perimeter is closed and adiabatic.

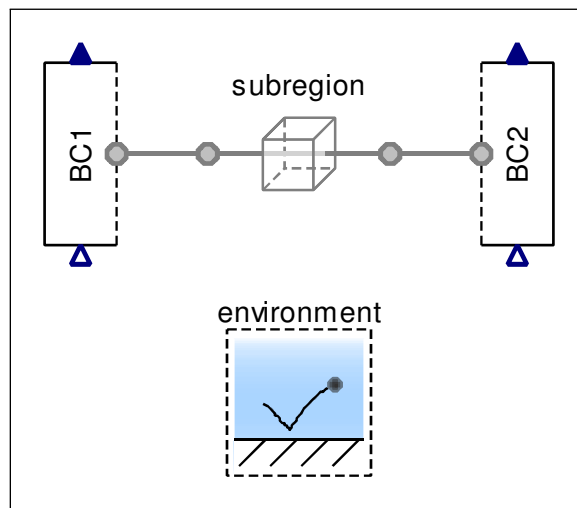
5.4.2 Results and Discussion

Modeling and simulation statistics:

- Number of variables: 374
- Number of time-varying variables: 31



(a) Physical domain.



(b) Model.

Figure 5.15: Configuration of the electrical conduction example.

- Number of states: 1
- Sizes of the nonlinear systems of equations: None
- Sizes of the linear systems of equations: None
- Translation time: 3 s
- Simulation time: 0.005 s

As expected, the simulation shows an electrical resistance of $R = L/\sigma A = 100\ \Omega$. The rate of heat generation is $\dot{Q}_{\text{gen}} = (zI)^2 R = 250\ \text{mW}$.

Figure 5.16 shows the temperature trend. As expected, the steady-state temperature is $T = T_0 + \theta L \dot{Q}_{\text{gen}}/4A \approx 82\ ^\circ\text{C}$, where T_0 is the boundary temperature ($25\ ^\circ\text{C}$), θ is the thermal resistance, L is the length (1 cm), and A is the cross-sectional area ($1\ \text{mm}^2$). The factor of one fourth is due to the boundary conditions; the conduction length is half of the total length and the heat is rejected to both sides. The time constant, as measured by the time to $1 - e^{-1}$ ($\approx 63\%$) of the final value, is approximately equal to $\tau_{QT}/2$, where τ_{QT} is the time constant for thermal conduction from one side into the subregion (Equation 3.208).

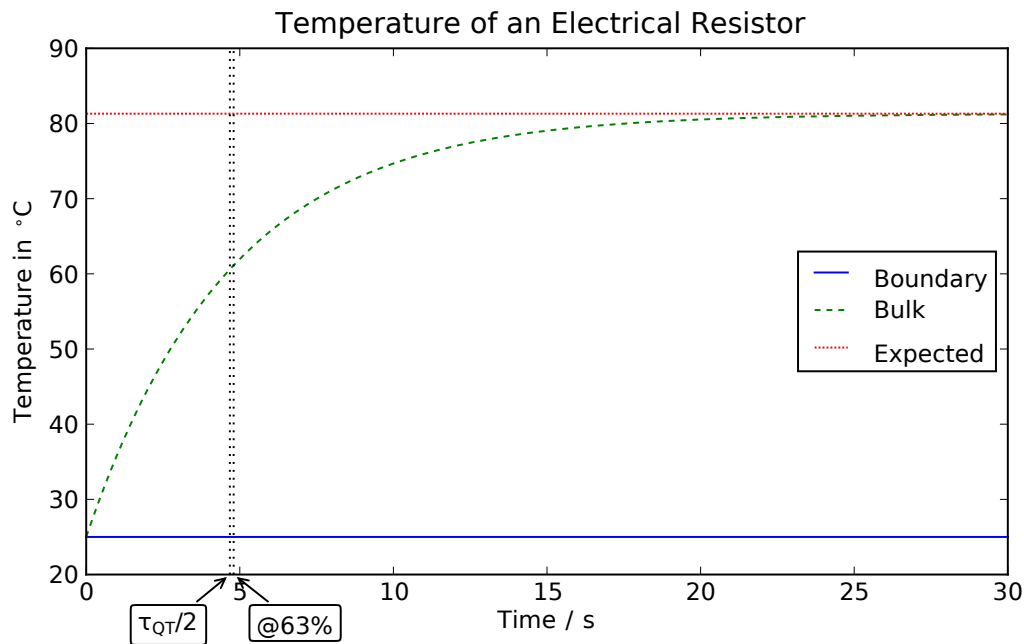


Figure 5.16: Dynamic heating of an electrical resistor.

5.5 Thermal Conduction

This example demonstrates distributed thermal conduction and heating. The key model equations are thermal conduction (Equation 3.145) and the energy balance (Equation 3.200).

5.5.1 Conditions

A graphite bar divided into eight 1 cm^3 subregions, as shown in Figure 5.17. The initial temperature of the first subregion is 55°C ; the others begin at 25°C . The outer boundaries are adiabatic.

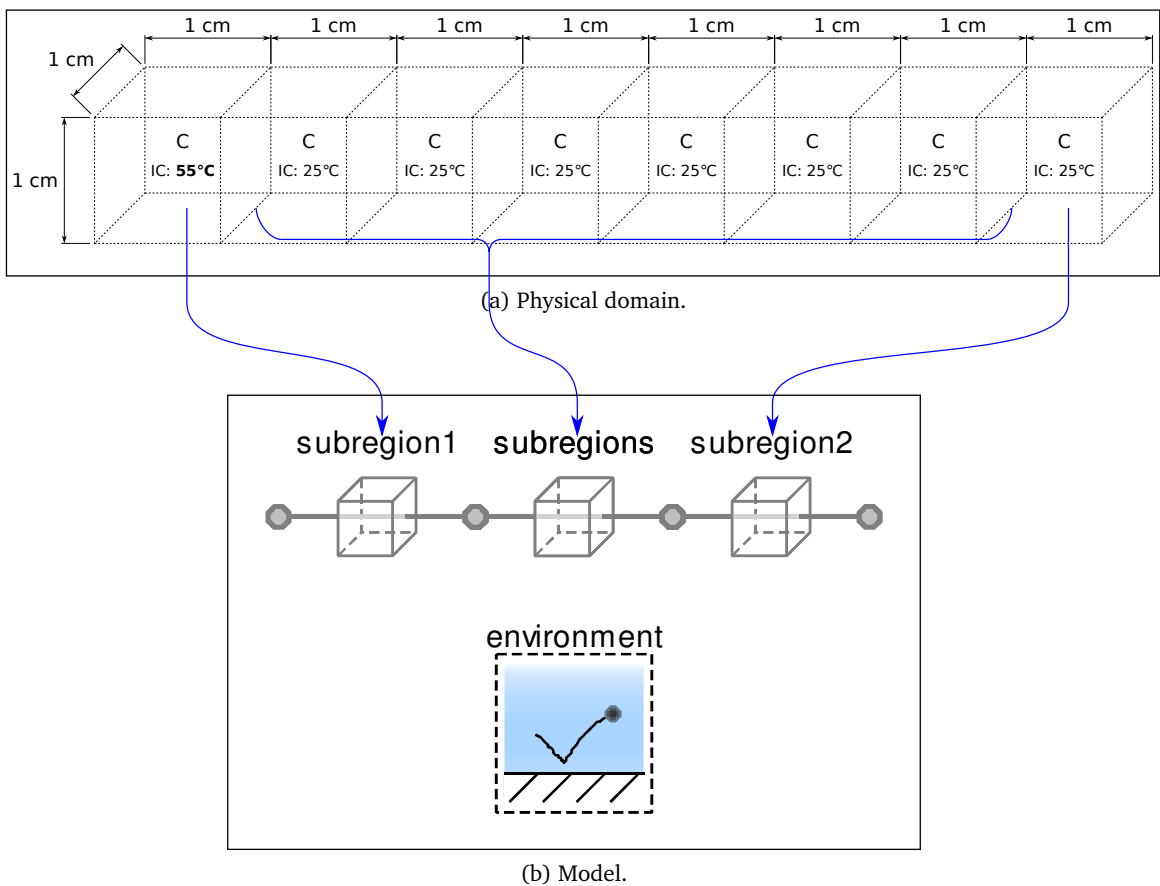


Figure 5.17: Configuration of the thermal conduction example.

5.5.2 Results and Discussion

Modeling and simulation statistics:

- Number of variables: 1119

- Number of time-varying variables: 80
- Number of states: 8
- Sizes of the nonlinear systems of equations: None
- Sizes of the linear systems of equations: 7 sets of 2
- Translation time: 3 s
- Simulation time: 0.009 s

Figure 5.18 shows the temperature trends throughout the graphite bar. The temperatures distribute evenly and then converge to a final temperature of approximately 28 °C after 500 s. Since there is no material transport, the temperatures of the interfaces are the average temperatures of the neighboring subregions.

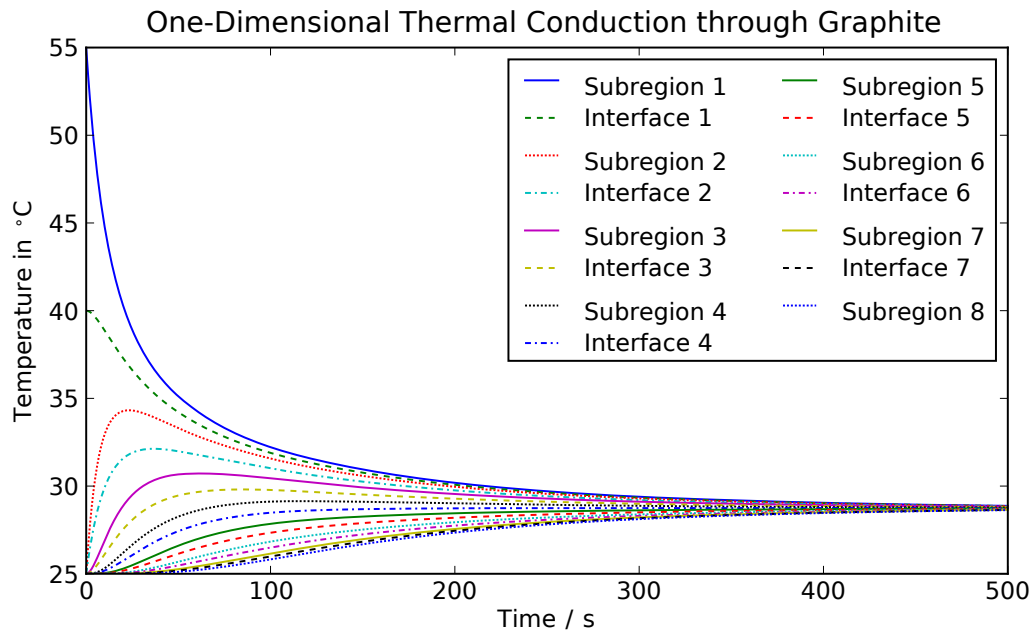


Figure 5.18: Temperature in a graphite bar during transient conduction.

5.6 Thermal Conduction and Convection

This example builds on the previous one. It adds gas to the system and shows how the gas moves as the temperature equalizes. Material transport (Equation 3.104) and translational exchange (Equation 3.55) are important here.

5.6.1 Conditions

The conditions are the same as described in Section 5.5.1, except that N_2 gas now occupies half of the volume of each subregion. The gas is initialized to 1 atm and the same temperature as the solid. Initially, it has zero velocity. The translational independence factor is ten ($k_{\phi_x} = 10$), which represents a porous media.

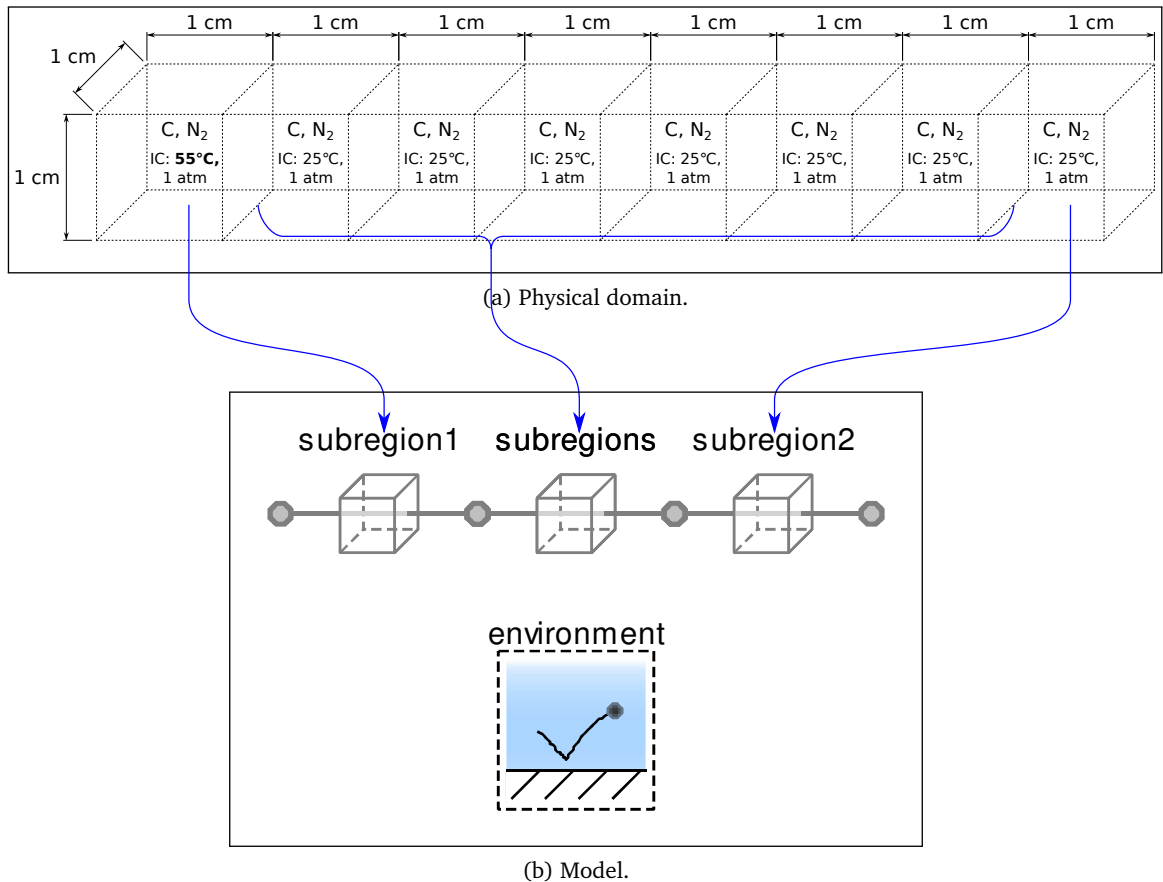


Figure 5.19: Configuration of the thermal conduction and convection example.

5.6.2 Results and Discussion

Modeling and simulation statistics:

- Number of variables: 2407
- Number of time-varying variables: 574
- Number of states: 31
- Sizes of the nonlinear systems of equations: None
- Sizes of the linear systems of equations: 43 sets of 2
- Translation time: 5 s
- Simulation time: 0.072 s

The simulation is approximately eight times slower than the example without the gas (Section 5.5). There are states for the temperature of the solid and the temperature and pressure of the gas in every subregion (24 states). There are also states for the x component of velocity in all but one subregion (7 states). One of the translational states is eliminated due to the closed outer boundaries.

The temperature trends are nearly the same as for the previous example (Figure 5.18). The gas and the solid remain at approximately the same temperature in each subregion.

Due to the equation of state, the gas in the first subregion becomes denser and has lower pressure as it becomes colder. This is shown in Figures 5.20 and 5.21. The gas in the first two subregions begins to travel very slowly (on the order of $10\ \mu\text{m/s}$) in the negative direction, into those subregions as they cool. Due to conservation of momentum, the gas begins to move in the opposite (positive) direction in other regions in the first 100 s. The expansion around the second interface is driven by a higher pressure there. Shortly after 100 s, the pressure in the last subregion reaches a maximum and counters the movement towards the positive outer boundary.

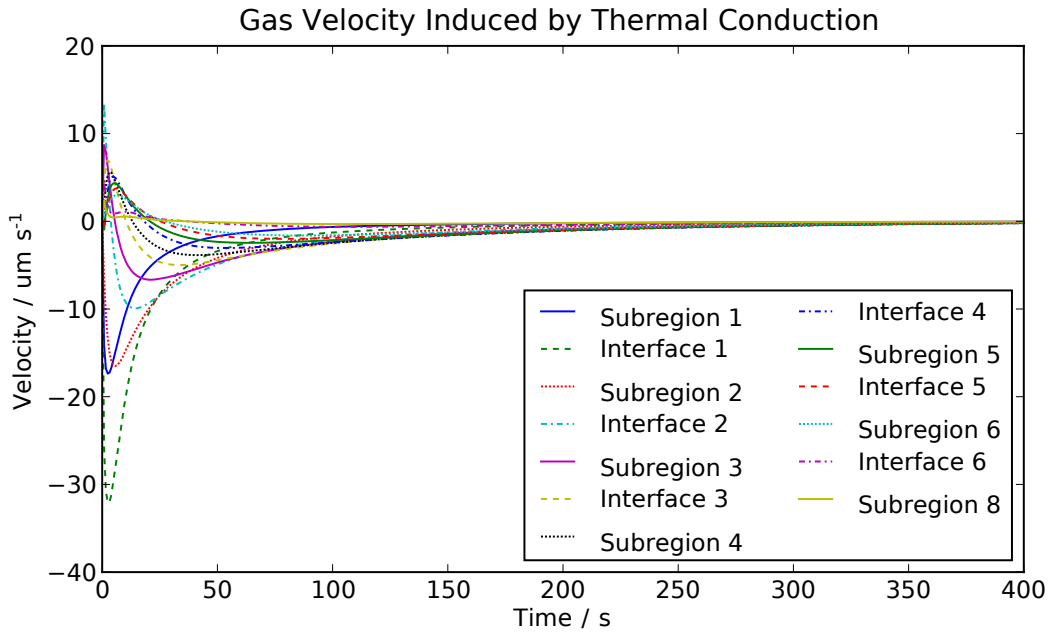


Figure 5.20: Velocity induced in gas in contact with graphite undergoing thermal conduction.

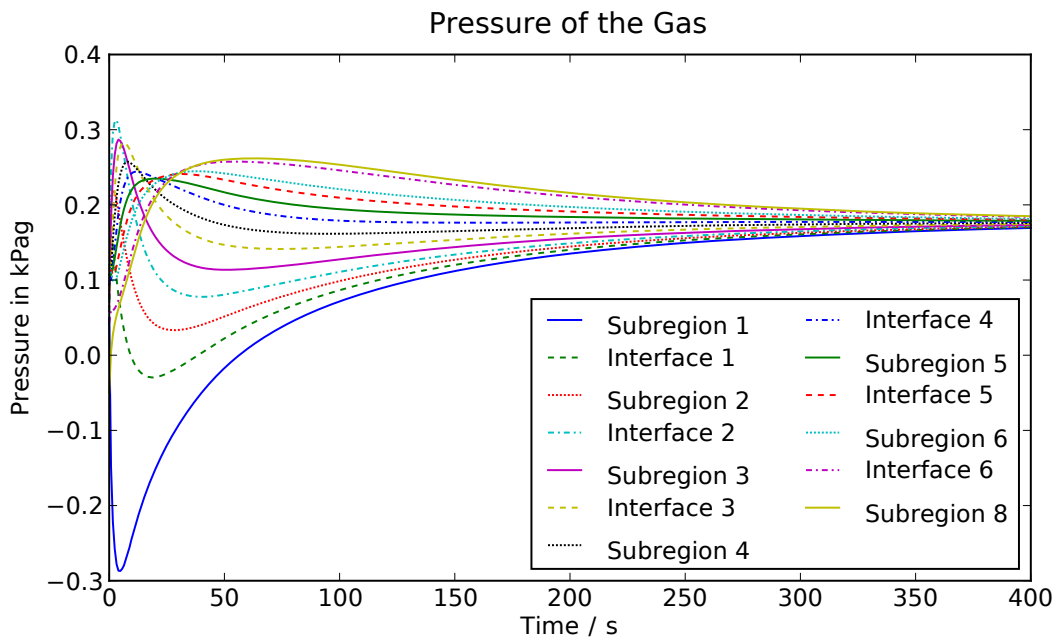


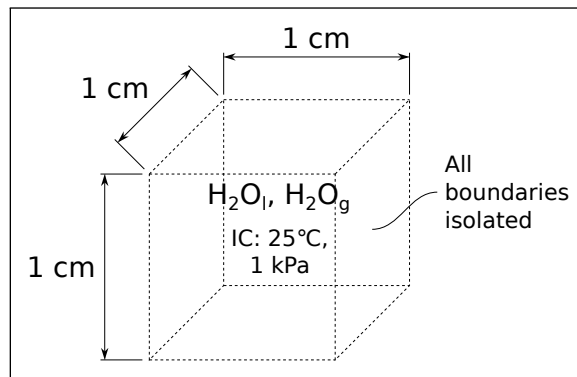
Figure 5.21: Pressure induced in gas in contact with graphite undergoing thermal conduction.

5.7 Evaporation

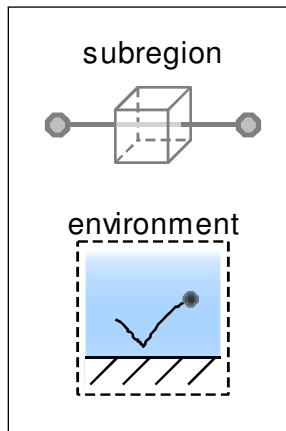
This example demonstrates the evaporation of liquid water into an initially sub-saturated vapor until equilibrium is reached. The cooling effect of evaporation is evident. The key equations are the rate of phase change (Equation 3.42) and the energy balance (Equation 3.200).

5.7.1 Conditions

Water vapor and liquid water are present in a closed, adiabatic volume of 1 cm^3 . Initially, 0.1% of the volume is filled with liquid, and the water is at 25°C and 1 kPa in both phases. These conditions are below saturation. Capillary pressure is considered negligible.



(a) Physical domain.



(b) Model.

Figure 5.22: Configuration of the evaporation and condensation example.

5.7.2 Results and Discussion

Modeling and simulation statistics:

- Number of variables: 537
- Number of time-varying variables: 143
- Number of states: 5
- Sizes of the nonlinear systems of equations: None
- Sizes of the linear systems of equations: 3, 2, 2
- Translation time: 3 s
- Simulation time: 0.012 s

Some of the liquid evaporates until saturation is reached. As shown in Figure 5.23, this takes approximately 2 ms. Figure 5.24 shows the rate of evaporation over time. The temperature is shown in Figure 5.25. It decreases by approximately 43 mK due to the latent heat of evaporation.

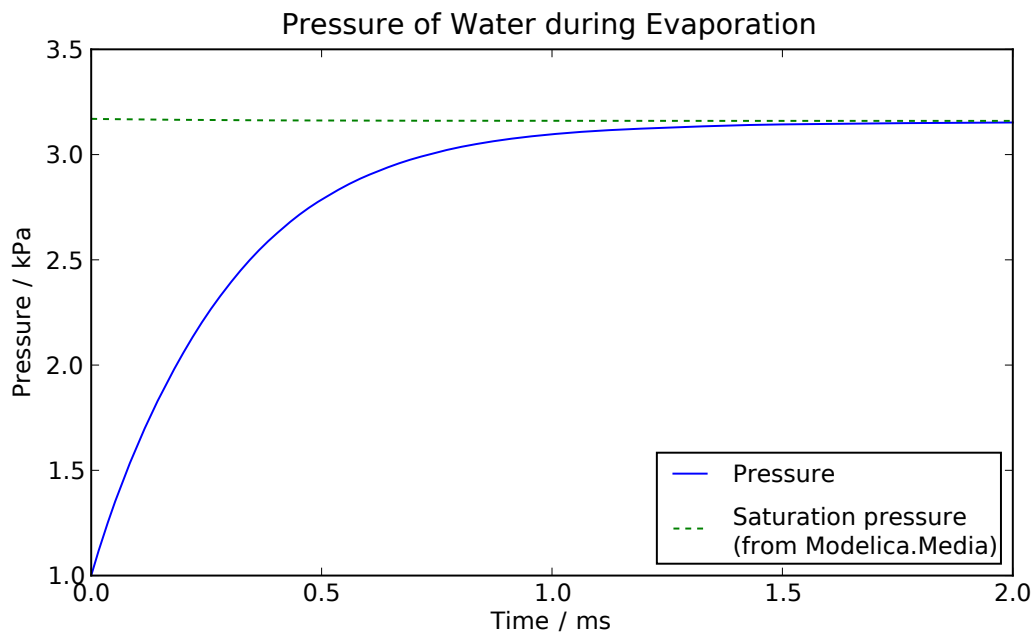


Figure 5.23: Pressure of H₂O reaching saturation.

As discussed in Section 3.5.1, the model uses thermodynamic properties (specific Gibbs energy) to describe phase equilibrium instead of a separate function for saturation pressure.

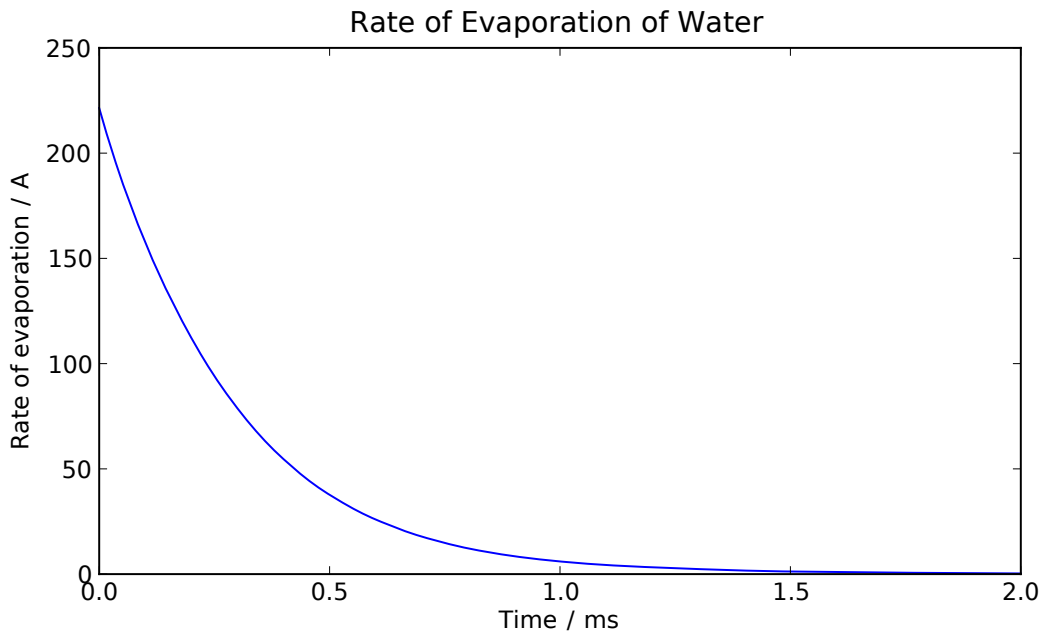


Figure 5.24: Rate of evaporation of initially sub-saturated H₂O.

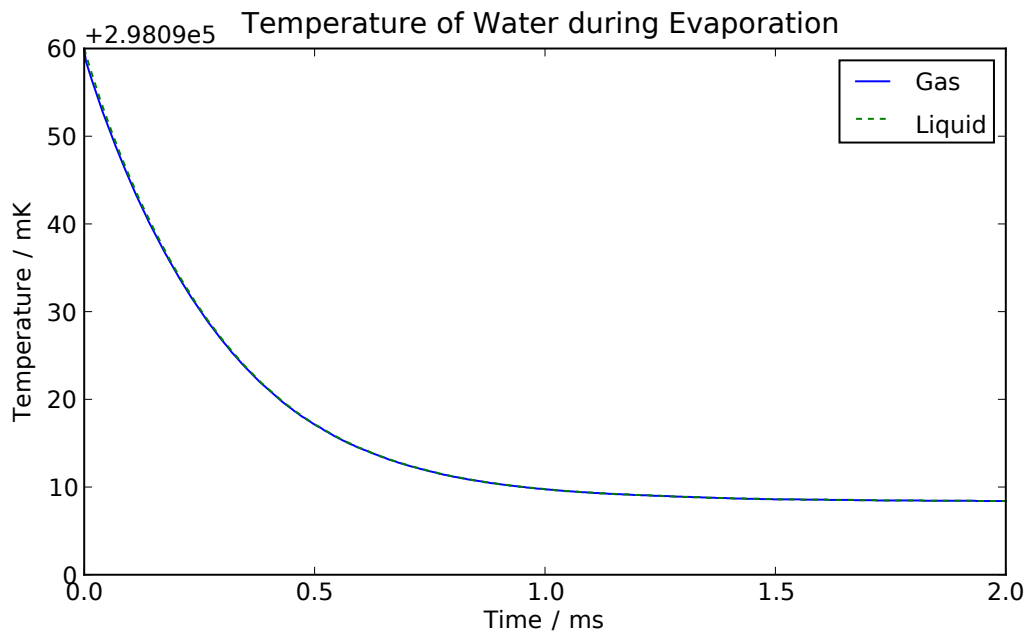


Figure 5.25: Temperature of H₂O during evaporation.

Figure 5.26 compares the equivalent saturation pressure of the model to a saturation pressure function from Modelica.Media. The model is evaluated as an ideal gas and using the second-order virial coefficients from [146]. The match is better with the second-order equation of state, but the model has sufficient accuracy with the ideal gas assumption.

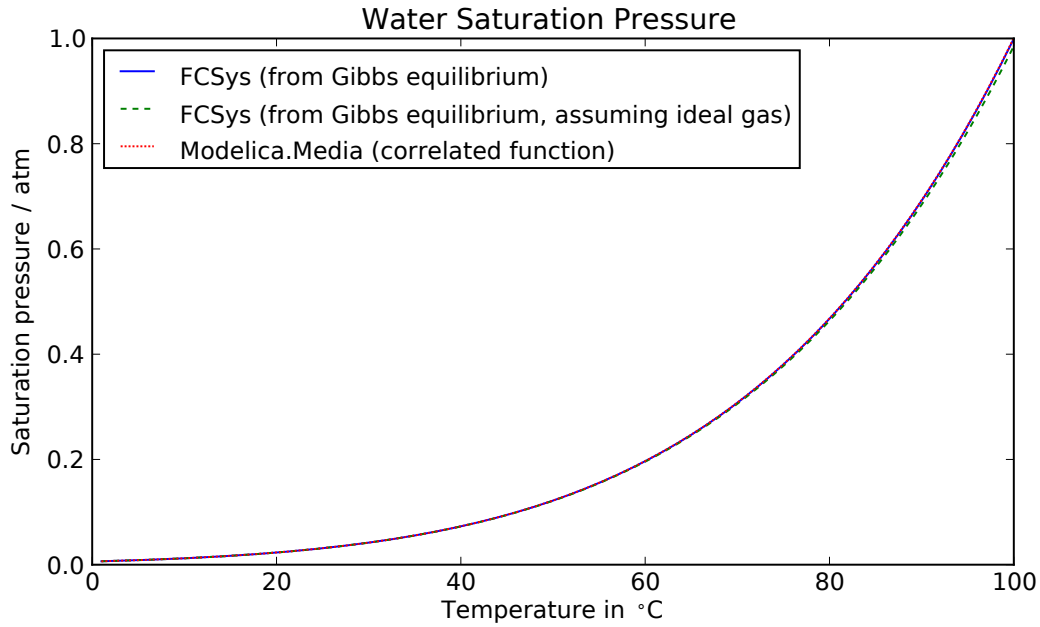


Figure 5.26: Validation of H₂O saturation pressure.

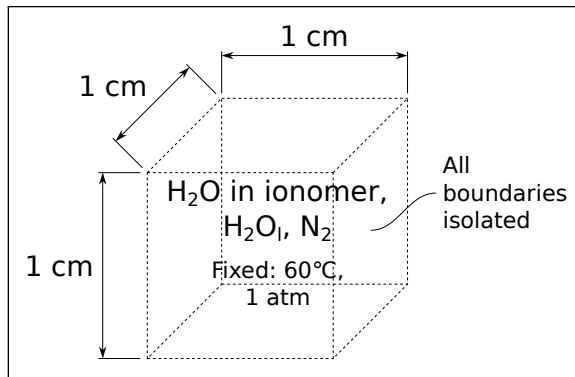
5.8 Hydration

This example is similar to the previous one, but the phase change is between the liquid and ionomer instead of between the liquid and gas.

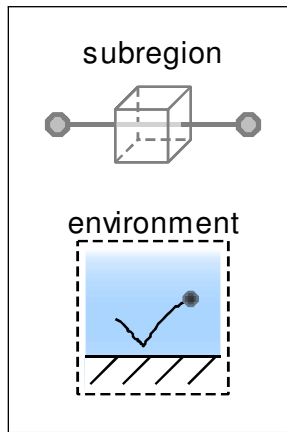
5.8.1 Conditions

Liquid and hydrated ionomer each occupy 30% of a cubic 1 cm³ control volume, as shown in Figure 5.27. No water enters or exits. An inert gas (N₂) fills the rest of the volume at 1 atm (fixed). In order to isolate the hydration, it is assumed that no evaporation takes place. The phase change interval (τ') is set to represent a case where the phases are interspersed with a relatively large amount of contact. All of the materials are held at 60°C. The ionomer is

initialized with a hydration level of $\lambda = 8$, where λ is the number of H_2O molecules divided by the number of SO_3^- end groups. This is below the equilibrium level of hydration.



(a) Physical domain.



(b) Model.

Figure 5.27: Configuration of the hydration example.

5.8.2 Results and Discussion

Modeling and simulation statistics:

- Number of variables: 654
- Number of time-varying variables: 68
- Number of states: 1
- Sizes of the nonlinear systems of equations: None
- Sizes of the linear systems of equations: 3 sets of 2
- Translation time: 3 s
- Simulation time: 0.011 s

Water is absorbed until the hydration level is in equilibrium with the liquid. This takes approximately two minutes, as shown in Figure 5.23. Figure 5.29 shows the rate of absorption over time.

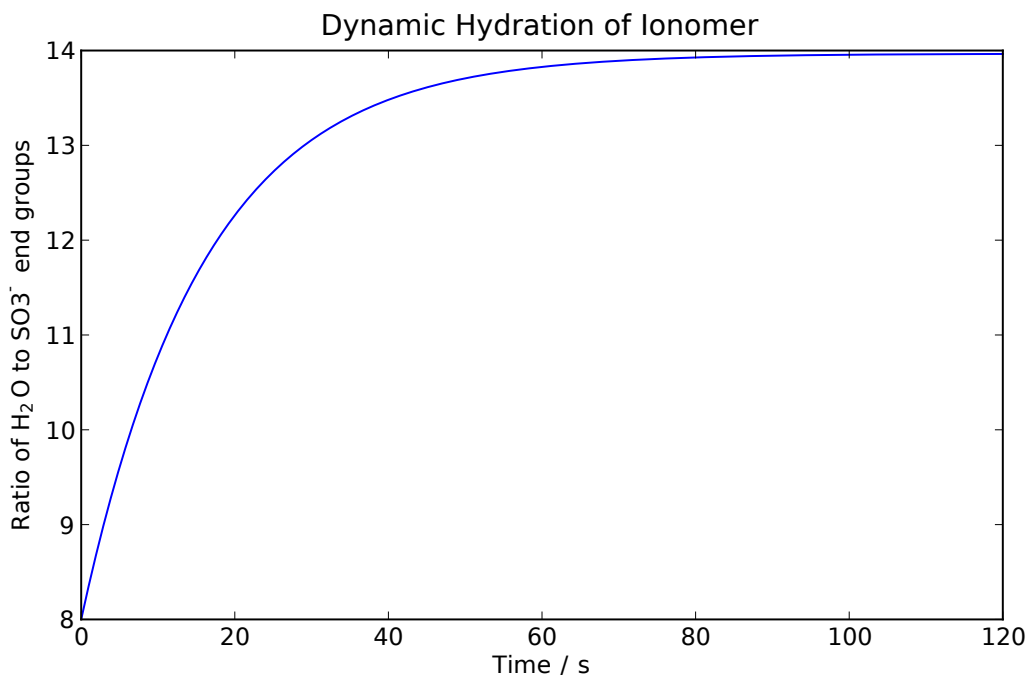


Figure 5.28: Hydration level of the ionomer reaching equilibrium.

The model uses thermodynamic properties (specific Gibbs energy) to describe phase equilibrium instead of a separate function for the equilibrium hydration level. Figure 5.30 compares the equivalent equilibrium hydration of the model at 30 °C to the explicit correlated function from Springer et al. [64]. As given by the implementation (Section 4.4), the model matches the correlation of Springer et al. [64] at 0 and 100%. Between these points, the relationship between relative humidity and hydration is linear and does not match the published correlation.

5.9 Summary

This chapter provided a wide array of basic examples of the model library. It is significant that all of the examples were built from the same model elements. They simulate quickly (≤ 72 ms; maximum in Section 5.6), and there are no nonlinear systems of equations. The

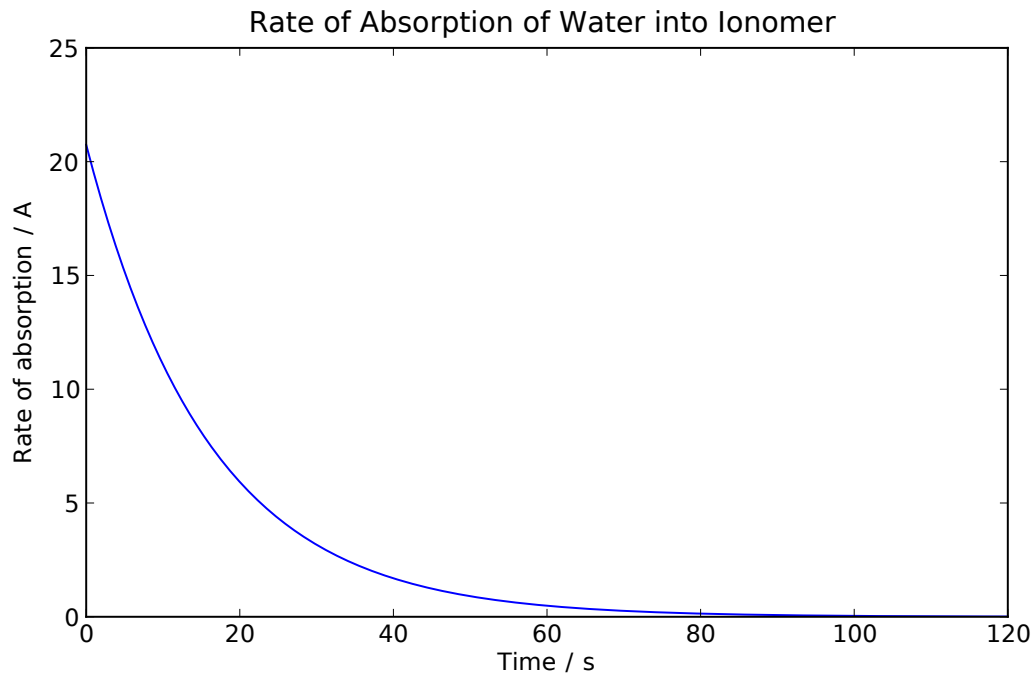


Figure 5.29: Rate of hydration of the ionomer.

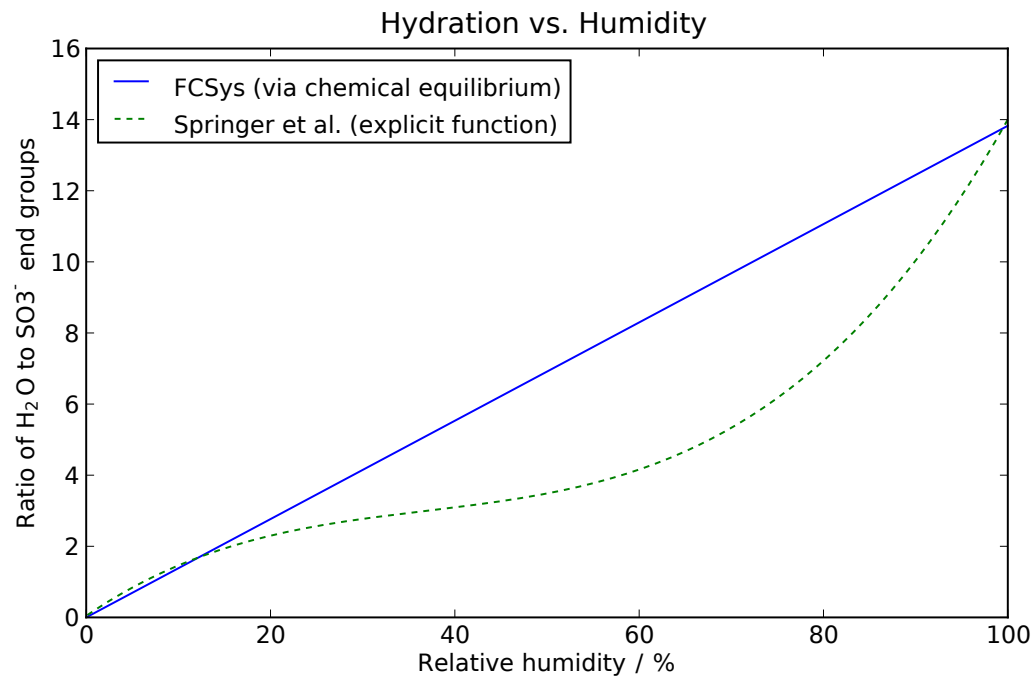


Figure 5.30: Equilibrium hydration level versus relative humidity.

settings of the phase change intervals in Sections 5.7 and 5.8 are the same as those used in the fuel cell model. The next chapter will address the fuel cell model.

SIMULATION OF THE FUEL CELL MODEL

This chapter presents the results of the fuel cell model. The first section investigates the operation of the cell during a baseline polarization test. The next six sections vary certain properties, such as temperature and pressure, and observe the effects on the polarization curve. Then, Sections 6.8 and 6.9 show variations of the cell model, specifically a simplified model and a model with multiple segments down the length of the channel. The final section (6.10) shows the dynamics of the model under a cyclical load. All of the results are from the fuel cell model except for the experimental data used in Figures 6.1 and 6.20 through 6.25 to benchmark the model.

6.1 *Baseline Polarization Test*

6.1.1 Test Conditions

The `TestStand` model presented in Section 4.12 applies boundary conditions to the `Cell` model. Under the baseline conditions, the temperature of the reactant supplies and the exterior x-axis boundary of the flow plates is 60 °C. This temperature is used to initialize the layers as well. Both outlets are maintained at 48.3 kPag. The cell operates on H₂ humidified to 80% and air humidified to 50%. The reactant supply varies stoichiometrically according to the electrical load. The stoichiometric ratio is 1.5 in the anode and 2.0 in the cathode. After an initial three minutes to reach steady conditions at 0.1 mA, the electrical load is ramped at a rate of 0.3 A/cm² per hour for ten hours or until the cathode is depleted of O₂. A slow ramp rate is used so that the dynamic effects are negligible. Although the model has options to assume steady state conditions, the states are generally necessary to avoid nonlinear systems of equations.

The area of the cell is 50 cm². There is one subregion for each layer. Ideal gases are assumed. Liquid is included. The double layer capacitances are not included in order to eliminate some of the dynamics. The key physical parameters of the model are listed in Table 6.1. The

values roughly correspond to a test cell used at the Hawaii Natural Energy Institute (HNEI) to provide the benchmark data, which is shown in the results. Where possible, the values are cited and justified in the model documentation (Appendix B). Due to the complexity of the model and the limitations of physical sensors, it is not possible to directly establish some of the parameters, especially those that deal with transfer rates. This is a known issue with fuel cell models in general [20].

Other parameters are listed in the tables of Appendix B. The thermodynamic correlations from McBride et al. [142] are used for the fluids. The fluidity and thermal resistivities are based on the correlations from Svehla et al. [178] and tables from Incropera and DeWitt [172]. The kinetic theory of gases [148] is used to approximate properties that are not available directly.

6.1.2 Results and Discussion

Modeling and simulation statistics:

- Number of variables: 6887
- Number of time-varying variables: 2749
- Number of states: 55
- Sizes of the nonlinear systems of equations: None
- Sizes of the linear systems of equations: 1 set of 9, 1 set of 8, 1 set of 6, 2 sets of 5, 9 sets of 4, 18 sets of 3, 97 sets of 2
- Translation time: 23 s
- Simulation time: 1.56 s

The cell model is much more complex than the models of Chapter 5, but as indicated by the simulation time of under two seconds, the translator and solver handle it well. There are no nonlinear systems of equations. The 55 states of the model include the following temperatures:

1. anFP gas
2. anFP liquid
3. anFP solid (graphite)
4. anGDL (all phases combined)
5. anCL (all phases combined)

Table 6.1: Key parameters of the fuel cell model.

	anFP	anGDL	anCL	PEM	caCL	caGDL	caFP
Volumetric porosity	0.059	0.8	0.4	—	0.4	0.8	0.042
Thickness / mm	10	0.235	0.029	0.04	0.029	0.235	10
Channel length / m	1.62	—	—	—	—	—	1.06
Hydraulic diameter / mm ²	0.94	—	—	—	—	—	0.82
Electronic conductivity / S cm ⁻¹	680	3.3 ^a	3.3 ^a	—	3.3 ^a	3.3 ^a	680
Protonic conductivity ^b / S cm ⁻¹	—	—	0.1	0.1	0.1	—	—
Thermal resistivity of carbon / m KW ⁻¹	0.01	0.85 ^a	0.85 ^a	—	0.85 ^a	0.85 ^a	0.01
Thermal resistivity of dry ionomer / m KW ⁻¹	—	—	6.25	6.25	6.25	—	—
Exchange current density @ 300 K / A cm ⁻²	—	—	1	—	2.3 × 10 ⁻⁵	—	—
Activation energy / V	—	—	0	—	0.75	—	—
Contact angle	90°	140°	140°	—	140°	140°	90°
Effective pore radius / μ m	—	32	5	—	5	32	—
Evaporation interval / ms	16	16	16	—	16	16	16
Absorption interval / s	—	—	33	—	33	—	—
Common mobility factor	10 ^{7c}	1.7	1.7	∞	1.7	1.7	10 ^{7c}
Mobility factor between gas and liquid	10 ^{6c}	∞	∞	—	∞	∞	10 ^{6c}
Mobility factor between H ⁺ and H ₂ O	—	—	0.02	0.02	0.02	—	—
Mobility factor between H ₂ O and SO ₃ ⁻	—	—	1	1	1	—	—
Common thermal independence factor	10 ⁵	0	0	0	0	0	10 ⁵

^aThis is for the bulk media (i.e., after accounting for porosity).

^bThis only describes coupling with the solid. Electro-osmotic drag (coupling with H₂O) introduces additional resistance.

^cThis is in the x direction (through the cell). In the y direction (down the channel), it is infinite.

6. PEM
7. caCL (all phases combined)
8. anGDL (all phases combined)
9. caFP gas
10. caFP liquid
11. caFP solid (graphite)

It is assumed that all of the species have the same temperature in the interior layers, regardless of phase, because the associated time constants were calculated by the model to be very quick (between 10^{-11} and 10^{-7} s). The following 24 amounts of material also appear as states:

1. H₂ in the anFP, anGDL, and anCL (3)
2. H₂O as gas in the anFP, anGDL, anCL, caCL, caGDL, and caFP (6)
3. H₂O as liquid in the anFP, anGDL, anCL, caCL, caGDL, and caFP (6)
4. H₂O in the ionomer of the anCL, PEM, and caCL (3)
5. N₂ in the caCL, caGDL, and caFP (3)
6. O₂ in the caCL, caGDL, and caFP (3)

There are 14 states related to velocity:

1. x-direction velocity of H₂ in the anGDL and anCL (2)
2. x-direction velocity of H₂O gas in the anGDL, caCL, and caGDL (3)
3. x-direction velocity of H₂O liquid in the anGDL, anCL, caCL, and caGDL (4)
4. y-direction velocity of H₂O liquid in the anFP and caFP (2)
5. x-direction velocity of H₂O in the ionomer of the anCL and caCL (2)
6. x-direction velocity of N₂ in the caGDL (1)

The y-direction velocities are constrained by the boundary conditions, so they are not independent for each species. The final group of states contains these currents:

1. x- and y-direction currents of water (H₂O) gas in the anFP (2)
2. x-direction current of oxygen (O₂) in the caCL and caGDL (2)
3. x- and y-direction currents of nitrogen (N₂) in the caFP (2)

For numerical reasons, it is sometimes better to choose currents as states instead of velocities.

Figure 6.1 compares the polarization of the cell model against seven runs of the experimental cell over the span of about a week. There are significant differences. At the lowest current measured by experiment (0.1 A/cm^2), the model overpredicts the voltage. This may be due to the hydrogen (H_2) crossover current, which was significant in the experimental cell but not included in the model. The modeling framework could be used to include this phenomena, but that development is left as future work. At mid-range current, the model underpredicts the voltage. This is a consequence of tuning the exchange current density to compromise between the low and mid-range current given the fact that H_2 crossover is not included. At higher currents, the voltage of the experimental cell drops—probably due to the effect of liquid water on the concentration/transport losses. Although liquid is included in the model and begins to fill the pores of the cathode gas diffusion layer (GDL) as will be shown, it does not have a significant effect on the polarization.

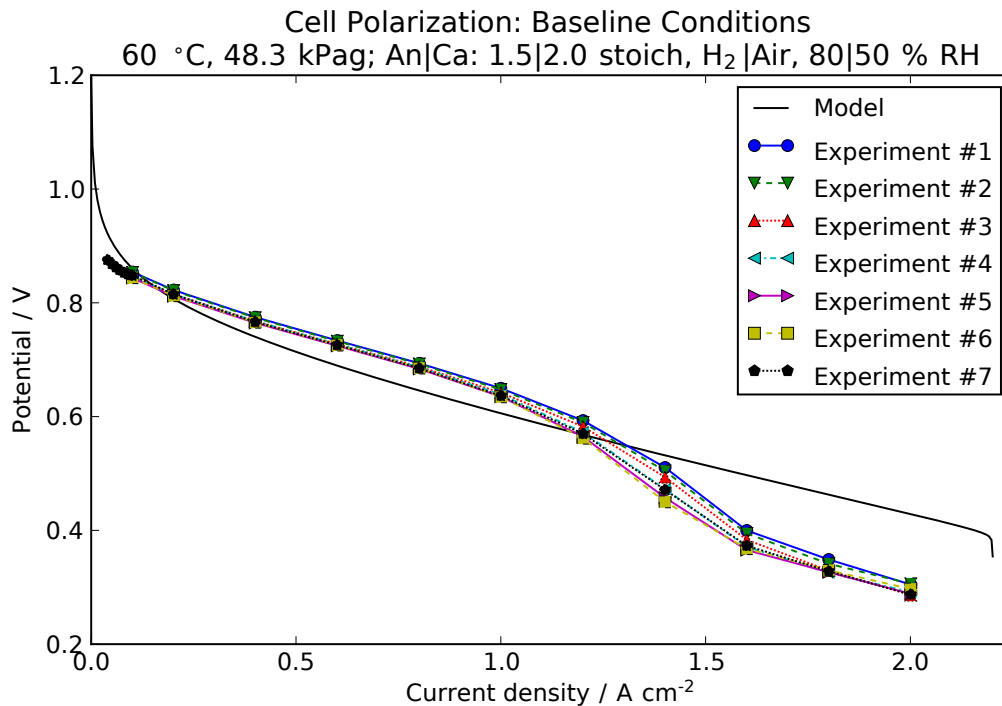


Figure 6.1: Polarization curves of the cell for the baseline polarization.

Figure 6.12 contains a Sankey diagram of the energy balance on the cell at 1.5 A/cm^2 . It should be noted that the Sankey diagram indicates the direction of transfer of the quantity

(here, energy), not necessarily the direction of material transfer. In Figure 6.12, the inlet and outlet of each stream is combined. While this helps to show the net rate of contribution, it may be confusing in the case of water. Since the enthalpy of formation of water is negative by convention, an inflow of water appears as an outflow of energy and vice versa.

The thermodynamic efficiency is only 36% at this rather high current of 1.5 A/cm^2 . The fact that the main trunk of the Sankey diagram has uniform thickness indicates that there is no transient energy storage (and the model equations are consistent). If the load is ramped quickly during a polarization test (e.g., in Section 6.10), then the energy balance includes significant transients. These would appear as a taper in the thickness of the trunk between the inputs and outputs.

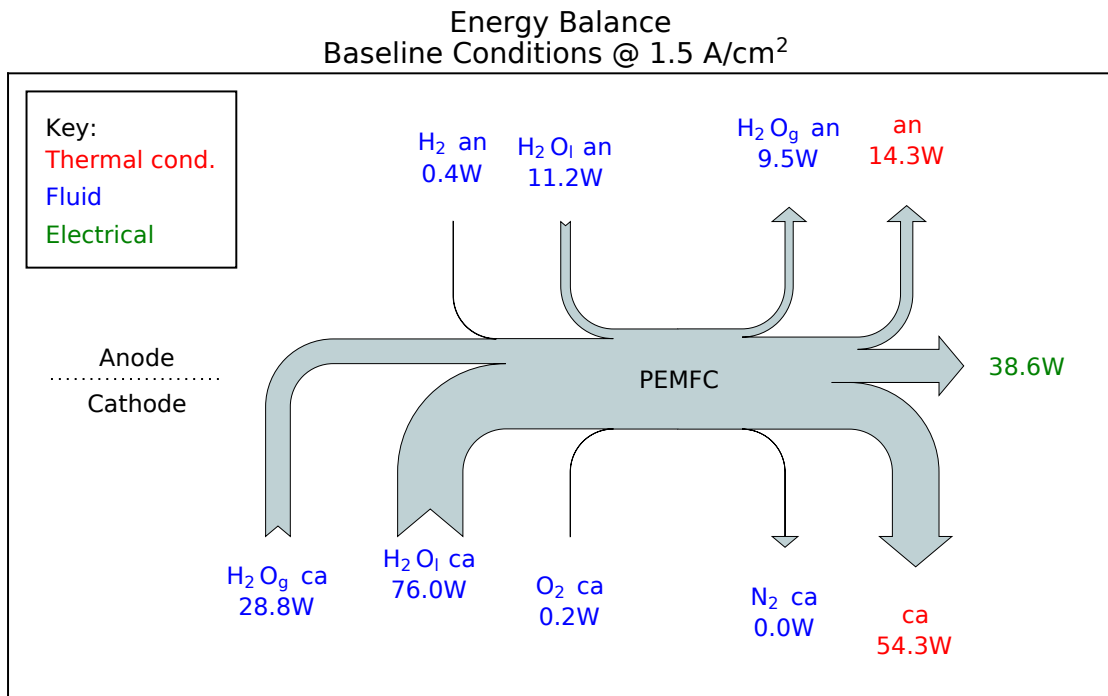


Figure 6.2: Energy balance under the baseline conditions at 1.5 A/cm^2 .

Figure 6.3 shows the losses in the cell. As expected, the cathode overpotential is dominant. The next most significant loss is protonic conduction through the catalyst layers and the proton exchange membrane (PEM). The anode operates in the regime where the overpotential is linear, in contrast to the Tafel or logarithmic/exponential regime of the cathode. The loss due to O_2

transport, while small in scale, does begin to increase exponentially near the limiting current density.

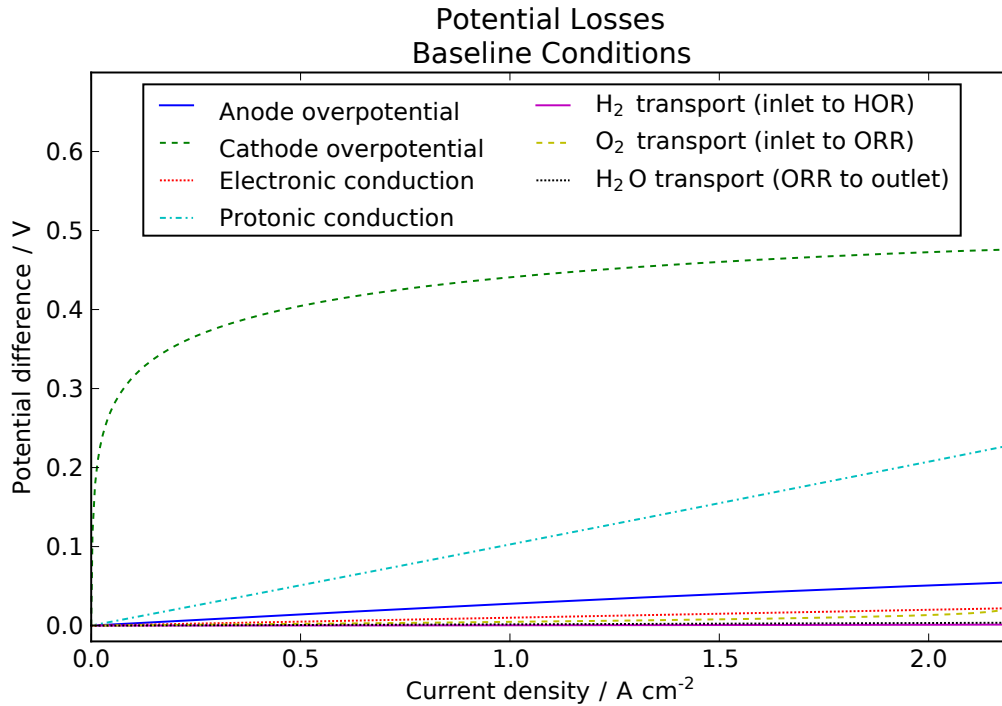


Figure 6.3: Potential losses through the cell during the baseline polarization.

Figure 6.4 gives information about location of the Ohmic losses, which were grouped as electronic and protonic in Figure 6.3. The loss in the PEM, which is protonic, is the most significant. The catalyst layers also have large losses, which are a combination of electronic and protonic. The resistance in the GDLs and flow plates is relatively small.

The losses generate heat and increase the temperature of the interior of the cell as shown in Figure 6.5. The hottest region is the cathode catalyst layer due to the large activation overpotential. The neighboring regions—the cathode GDL and the PEM—have the next highest temperatures. The cathode is generally hotter than the anode. More heat is rejected through the cathode flow plate, as was evident in Figure 6.2.

Figure 6.6 shows the pressure of the gas throughout the cell. It is highest in the cathode flow plate, as this is required to transport O₂ into the cell. Above 2 A/cm², the pressure of the cathode gas begins to decrease as the concentration limit is reached. The pressures of the anode are very close to the outlet pressure of 48.3 kPag. Figure 6.7 shows the pressure loss along the

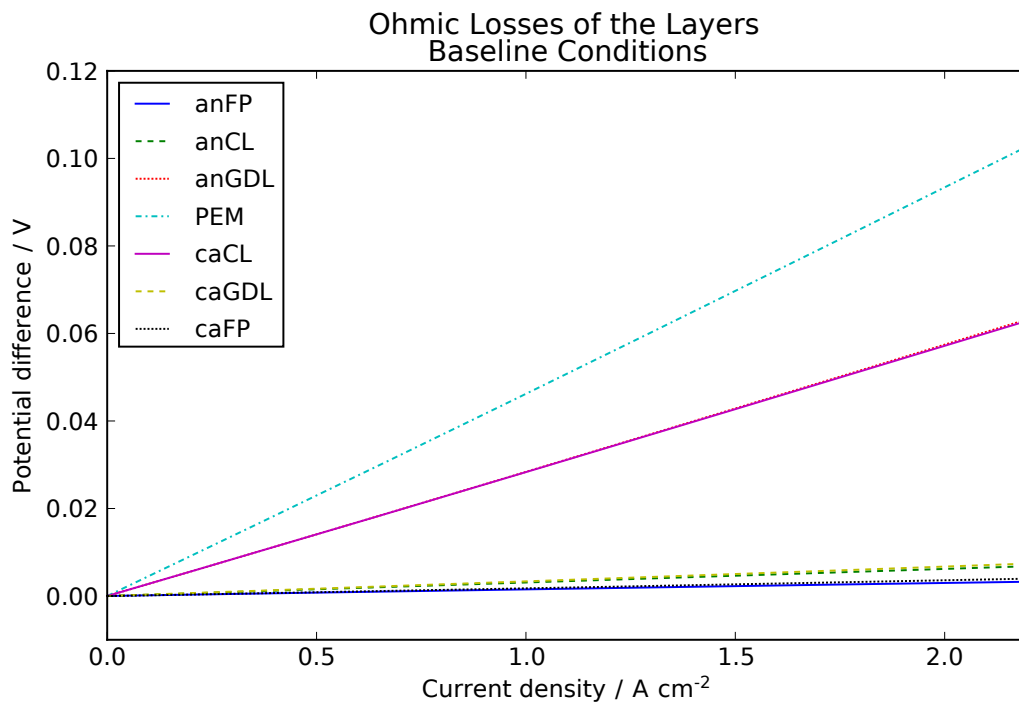


Figure 6.4: Electrical losses in the cell during the baseline polarization.

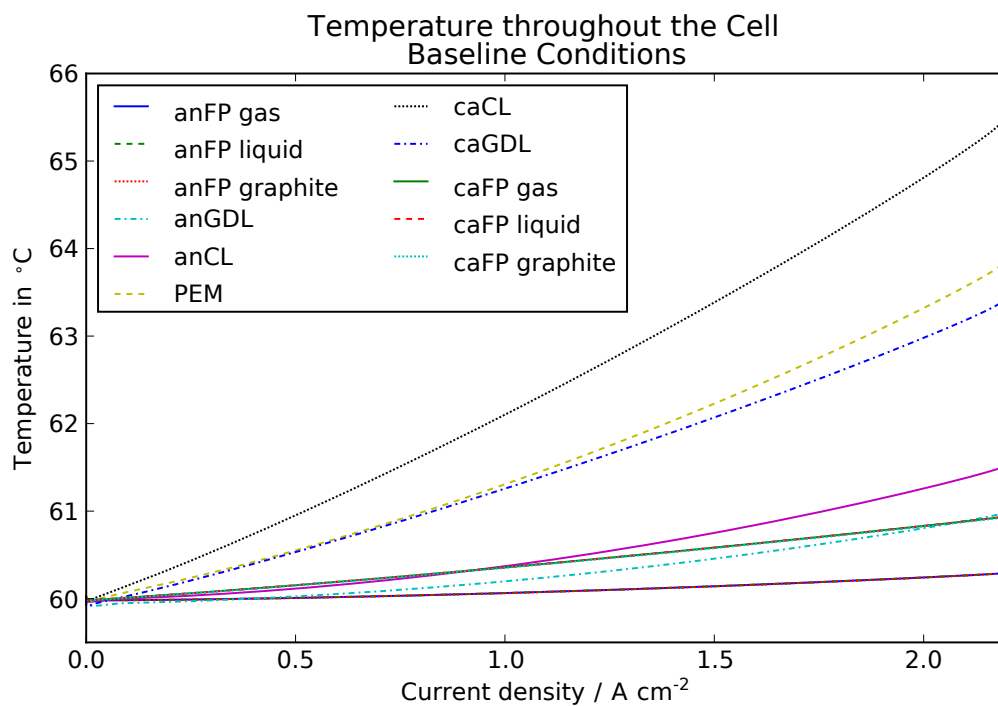


Figure 6.5: Temperatures through the cell during the baseline polarization.

channels. The pressure difference between the cathode inlet and outlet is much greater (a factor of nearly 20) than the difference between the anode inlet and outlet. This is due to the higher viscosity of air relative to H_2 and the higher velocity required to deliver O_2 since it is at lower concentration in the cathode than H_2 is in the anode.

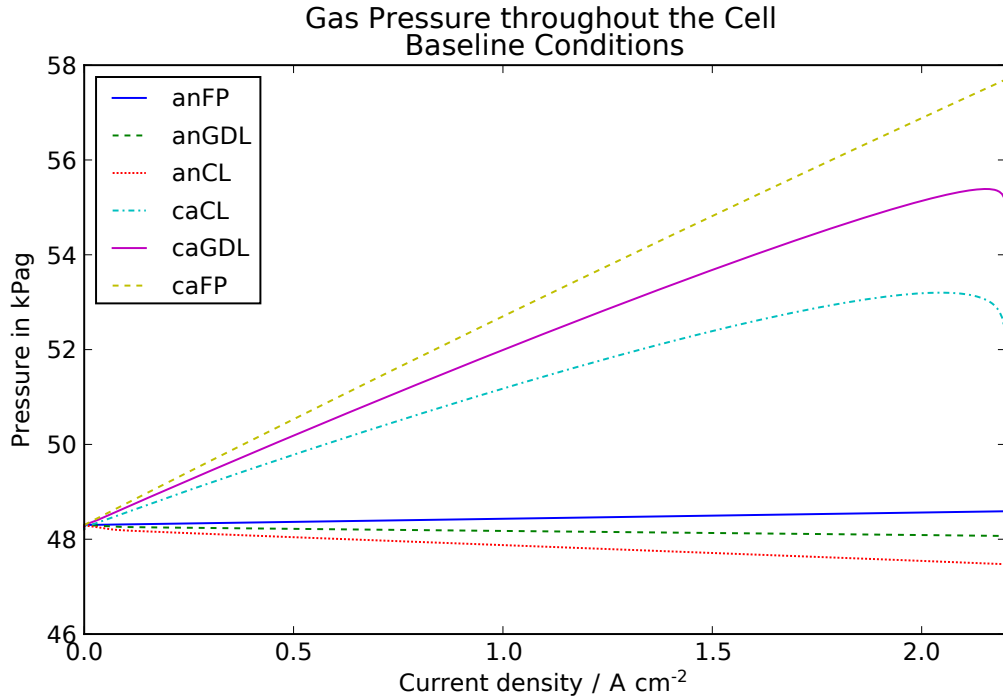


Figure 6.6: Total pressure of the gas during the baseline polarization.

Figure 6.8 shows the flow through the channels as indicated by the velocity of H_2O . The velocity is approximately five times higher in the cathode. In both channels, the velocity of the gas is much higher than the velocity of the liquid. The gas drags the liquid, yet the liquid has higher viscosity.

The model also includes the pressures of individual gas species. Figures 6.9, 6.11, and 6.10 show the pressures of H_2 , O_2 , and H_2O along their primary transport paths. The pressure of H_2 decreases from the anode inlet to the hydrogen oxidation reaction (HOR) and the pressure of O_2 decreases from the cathode inlet to the oxygen reduction reaction (ORR). The pressure of H_2O drops from the ORR to the cathode outlet.

Not all of the H_2O exits from the cathode. The Sankey diagram in Figure 6.12 shows the water balance at $1.5\ A/cm^2$. Water is generated at the rate of $37.5\ A/cm^2$ ($1.5\ A/cm^2\ e^- \times$

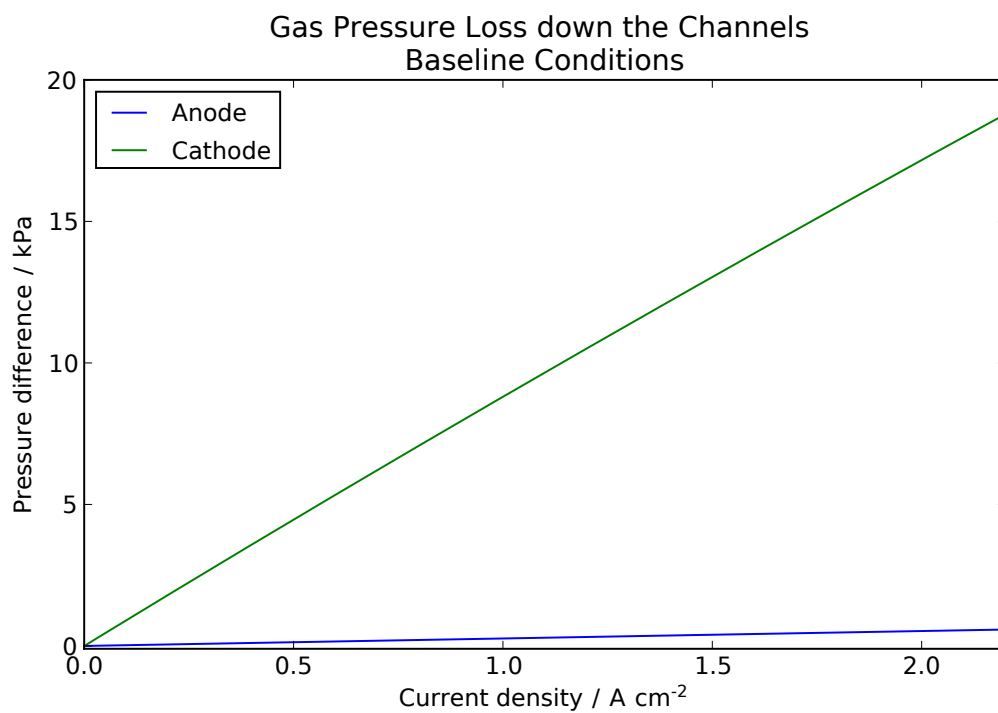


Figure 6.7: Pressure drop down the flow channels during the baseline polarization.

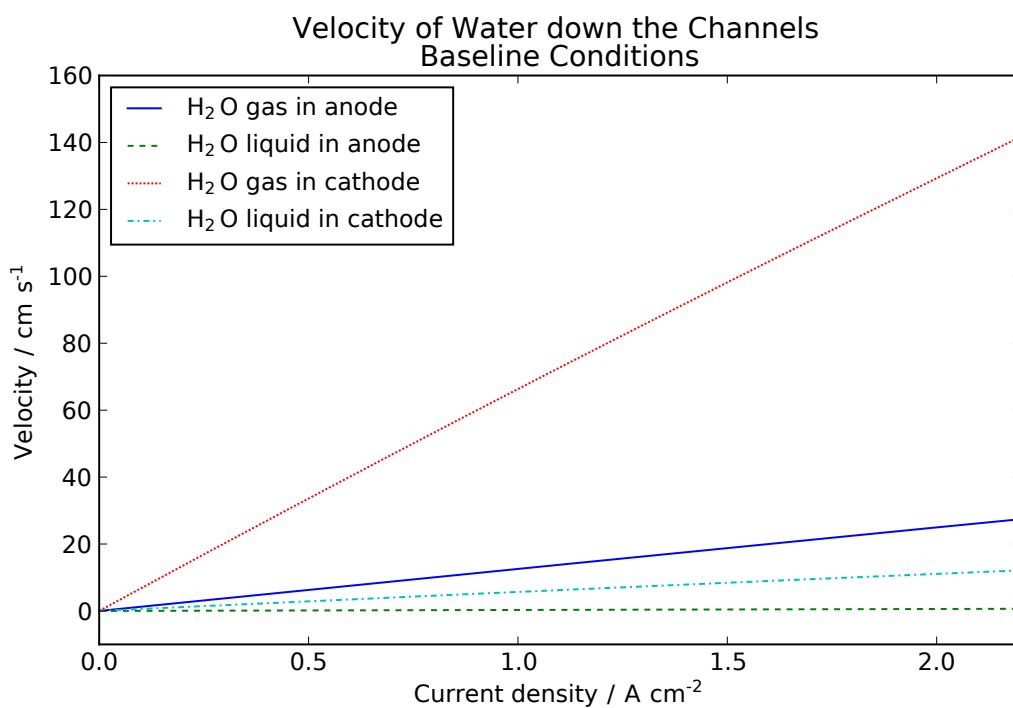


Figure 6.8: Velocity of H₂O down the channels during the baseline polarization.

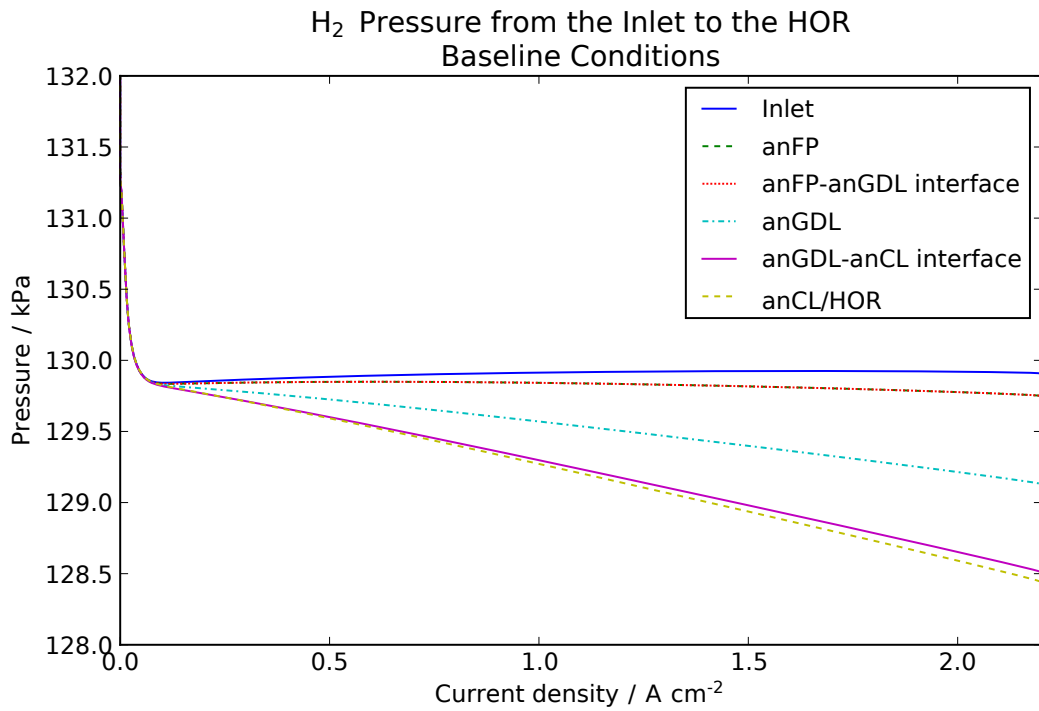


Figure 6.9: H₂ pressure during the baseline polarization.

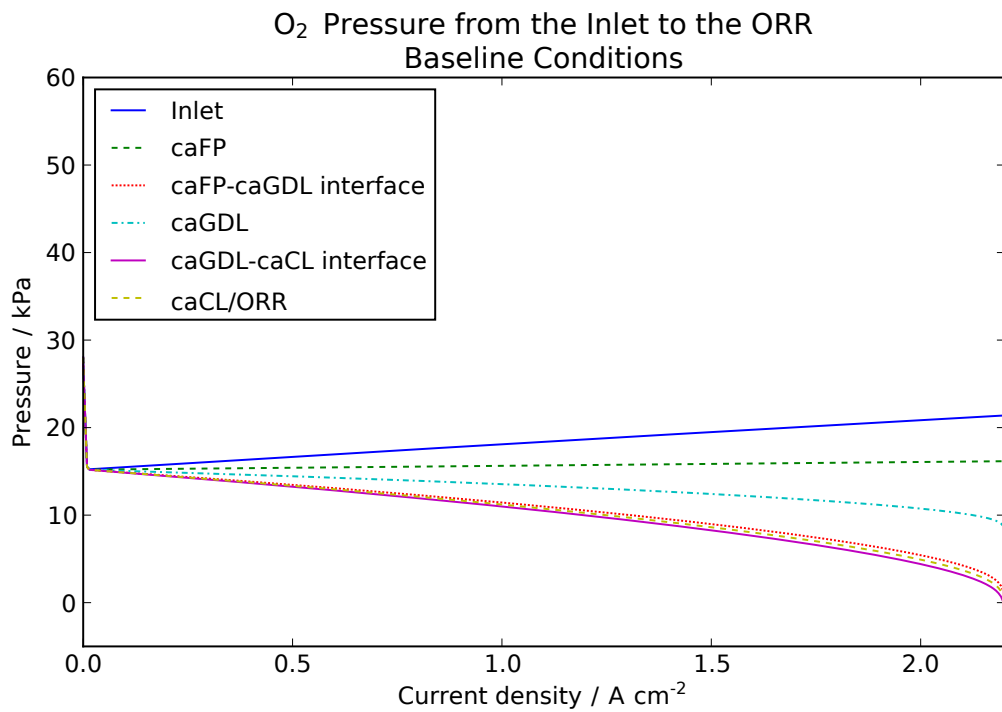


Figure 6.10: O₂ pressure during the baseline polarization.

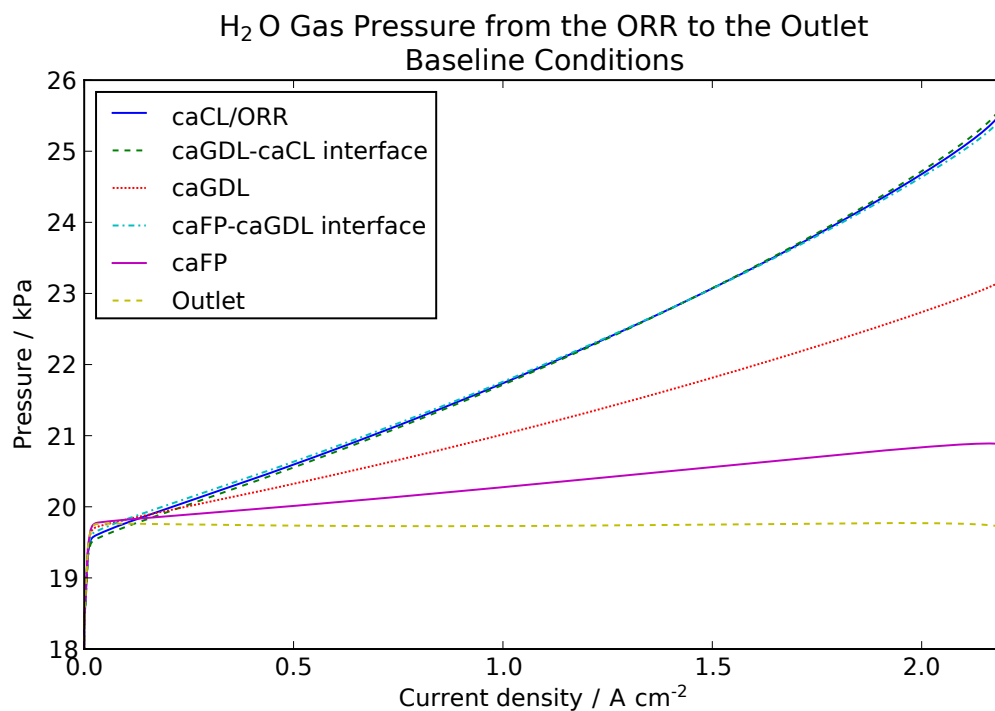


Figure 6.11: H₂O pressure during the baseline polarization.

$50 \text{ cm}^2 \times 1 \text{ H}_2\text{O}/2e^-$). There is a net input of water vapor into the anode and a net output of liquid. Water exits the cathode channels as gas and liquid.

Figure 6.13 shows the transport of water through the layers. It condenses near the interface of the anode flow plate and the anode GDL because it is super-saturated in the GDL relative to the channel due to the GDL's hydrophobicity. The current of water through the ionomer is relatively small, but as shown in Figure 6.14, it is not zero. At low currents ($< 1 \text{ A}/\text{cm}^2$), there is a fairly linear increase in the rate of transport through the PEM from the anode to the cathode due to the concentration gradient generated by the production of water in the cathode. However, the electro-osmotic drag becomes significant at high electrical currents and counters the diffusion. At the limiting current density, there is a rapid reversal of the trend caused by electro-osmotic drag because the cathode fills with liquid.

Figure 6.15 shows the level of hydration in the ionomer of the catalyst layers and the PEM. As the electrical current increases, water is produced more rapidly. This increases the hydration, beginning in the cathode catalyst layer. Although the concentration gradient is larger at higher

Water Balance
Baseline Conditions @ 1.5 A/cm²

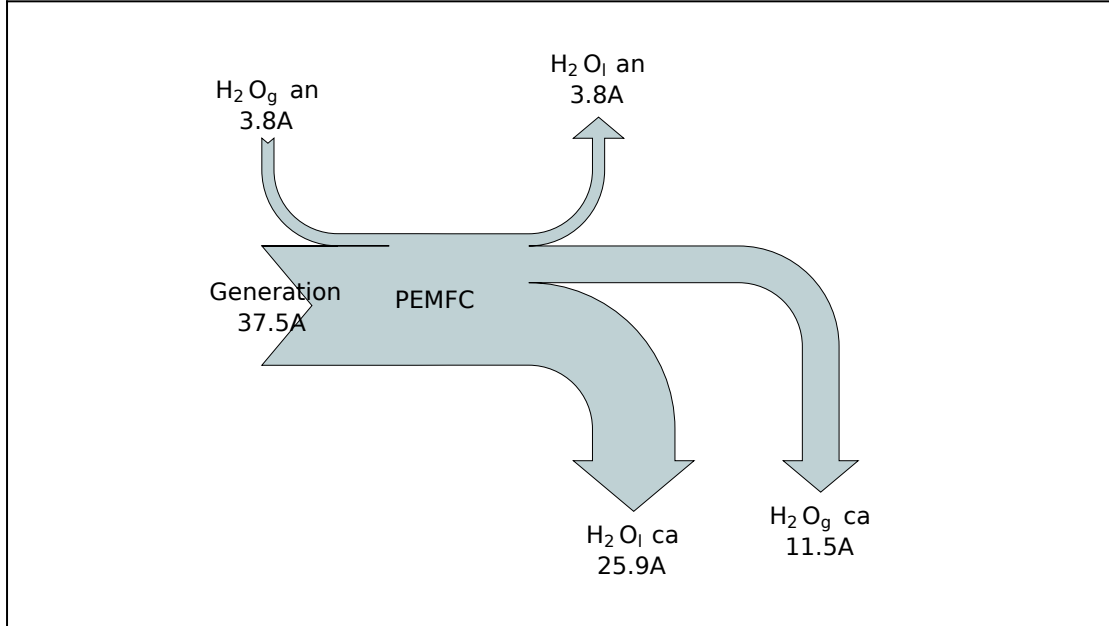


Figure 6.12: H₂O balance under the baseline conditions at 1.5 A/cm².

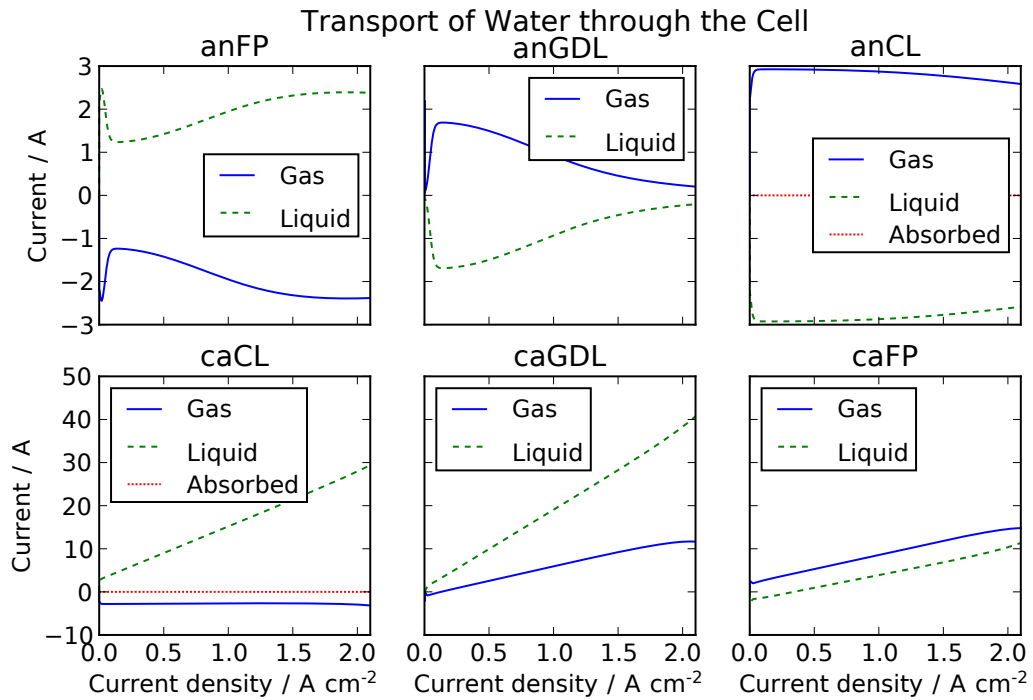


Figure 6.13: H₂O transport during the baseline polarization.

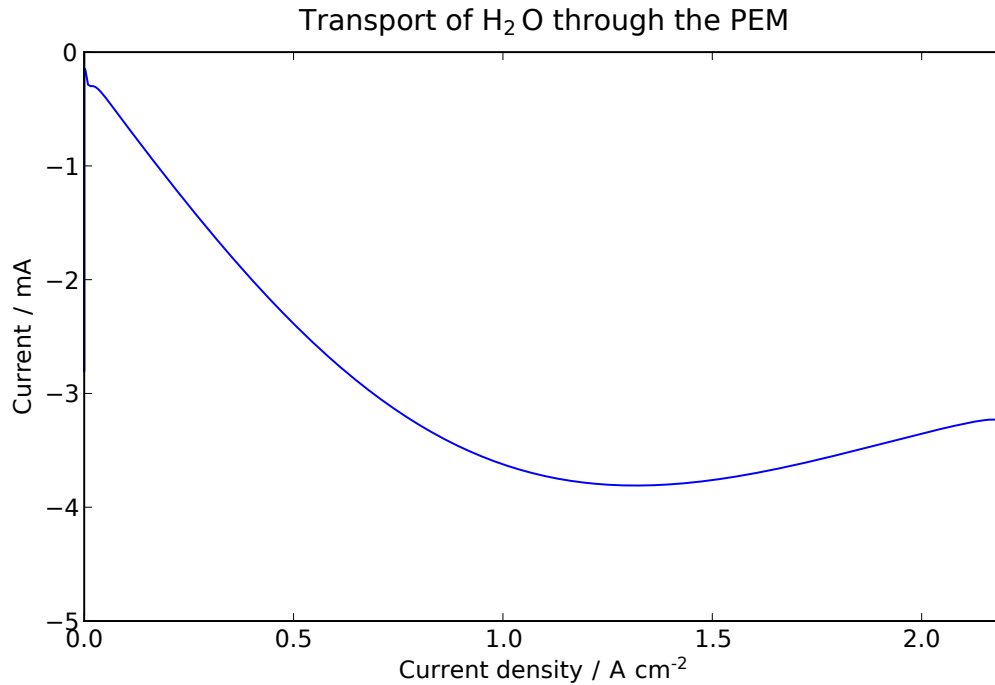


Figure 6.14: H₂O transport through the PEM during the baseline polarization.

electrical currents, so is the drag due to protonic transport. As was seen in Figure 6.14, the diffusion reaches a maximum (in magnitude) around 1.3 A/cm².

Besides entering the membrane, water also fills the pores of the cathode catalyst layer and GDL. This is shown in Figure 6.16. At the limiting current density, the GDL pores are 45% filled and rapidly filling further. The catalyst layer retains water to a lesser extent. The model assumes that the pores are smaller in the catalyst layer than the GDL (see Table 6.1) but have the same hydrophobicity. This yields a stronger capillary pressure, which tends to prevent flooding in the catalyst layer.

Figure 6.17 shows the humidity throughout the cell. The relative humidity is slightly higher than 100% in the cathode channel because the water enters the channel from the catalyst layer and the GDL, which are hotter due to heat generation. Also, the GDL is modeled as hydrophobic, which implies a lower saturation pressure (i.e., the Kelvin equation) such that saturated vapor in the GDL is supersaturated in the channel (even at the same temperature). The vapor in the anode flow plate is saturated, but it appears slightly below 100% relative humidity. The reason is that the relative humidity is only an output/analysis variable in the

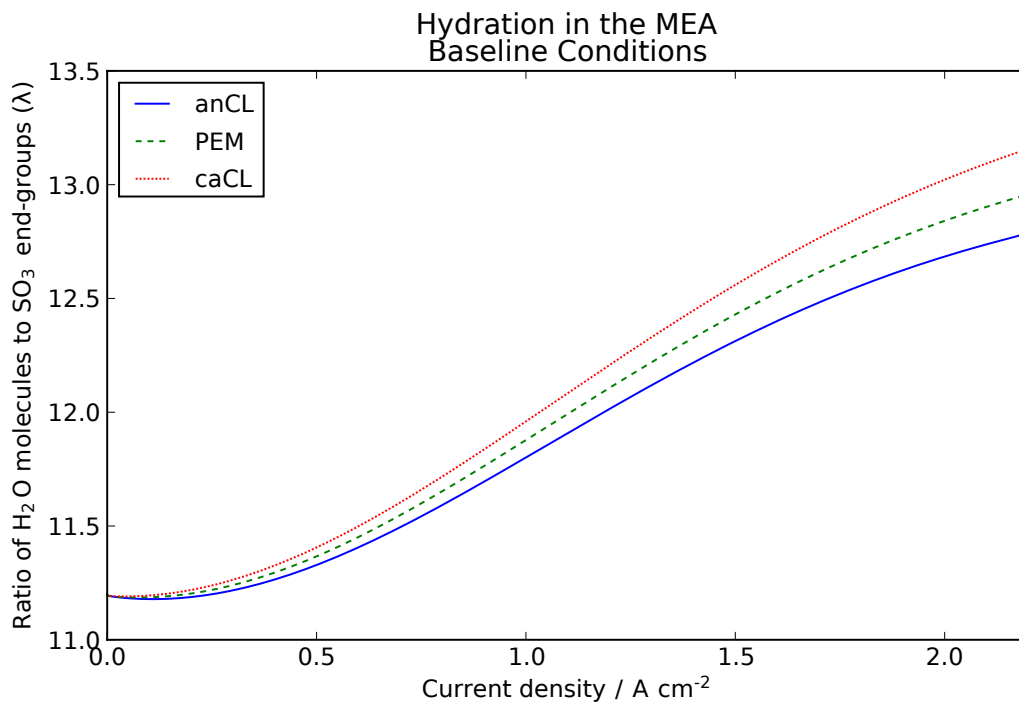


Figure 6.15: Hydration of the MEA during the baseline polarization.

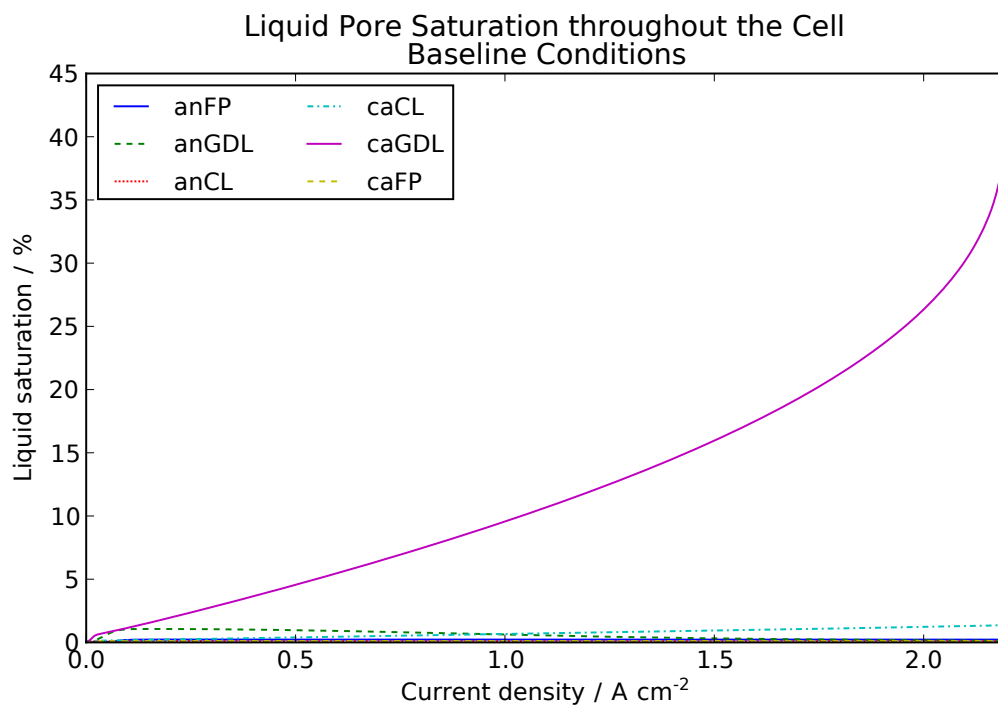


Figure 6.16: Liquid pore saturation during the baseline polarization.

model. It is calculated from `Modelica.Media.Air.MoistAir.saturationPressure()`, which does not correspond exactly to the model. This was apparent in Figure 5.26.

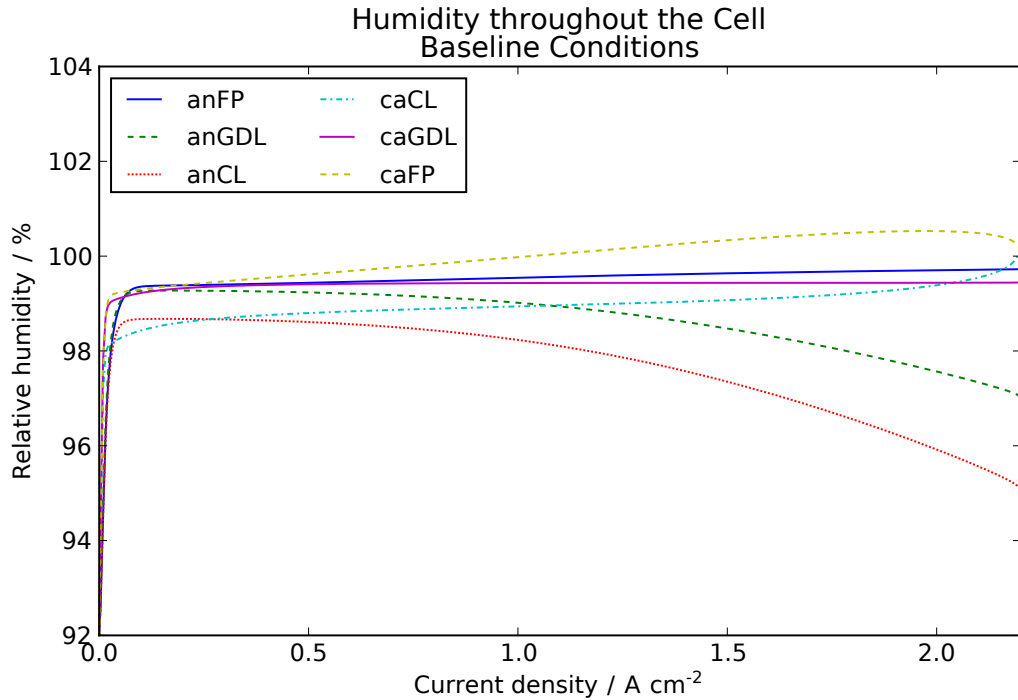


Figure 6.17: Relative humidity through the cell during the baseline polarization.

Figure 6.18 shows the rates of evaporation throughout the cell. Water evaporates most rapidly in the cathode catalyst layer. The ORR is modeled as producing liquid, not vapor, and the liquid evaporates to reach saturation. Water condenses in the other layers. The largest rate is in the cathode channel.

Figure 6.19 shows the rates of hydration in the catalyst layers. Water is absorbed on the cathode side and desorbed on the anode side where the relative humidity is lower (see Figure 6.17) and there is evaporation (see Figure 6.18). Note, however, that the rate of hydration is much slower than the rate of evaporation. This is consistent with the examples in the last two sections of Chapter 5.

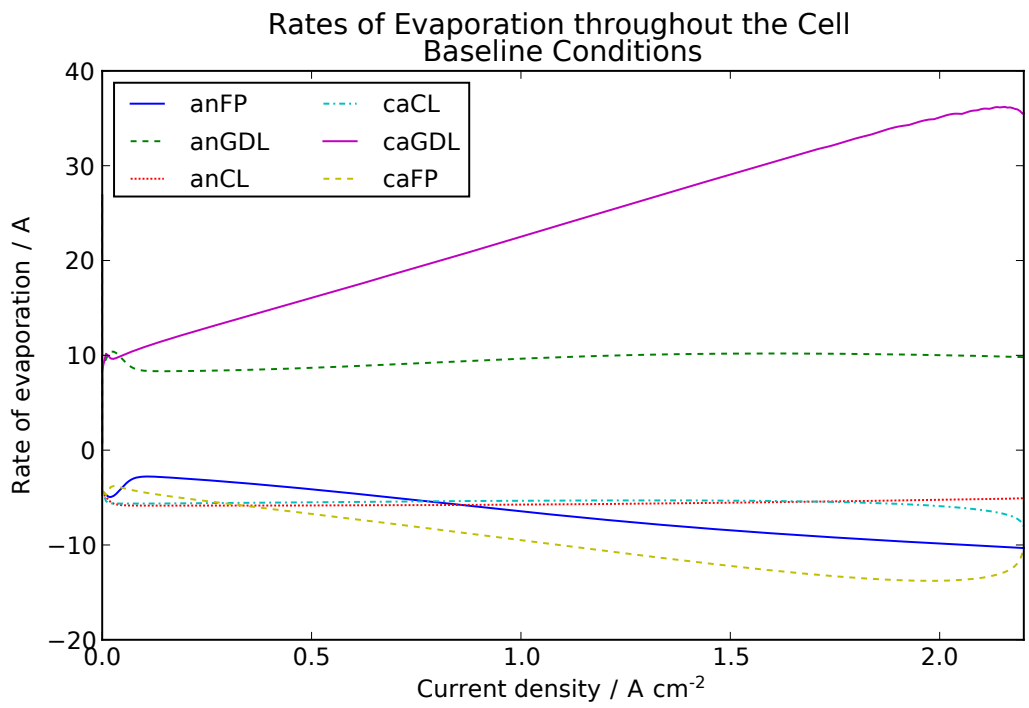


Figure 6.18: Rates of evaporation during the baseline polarization.

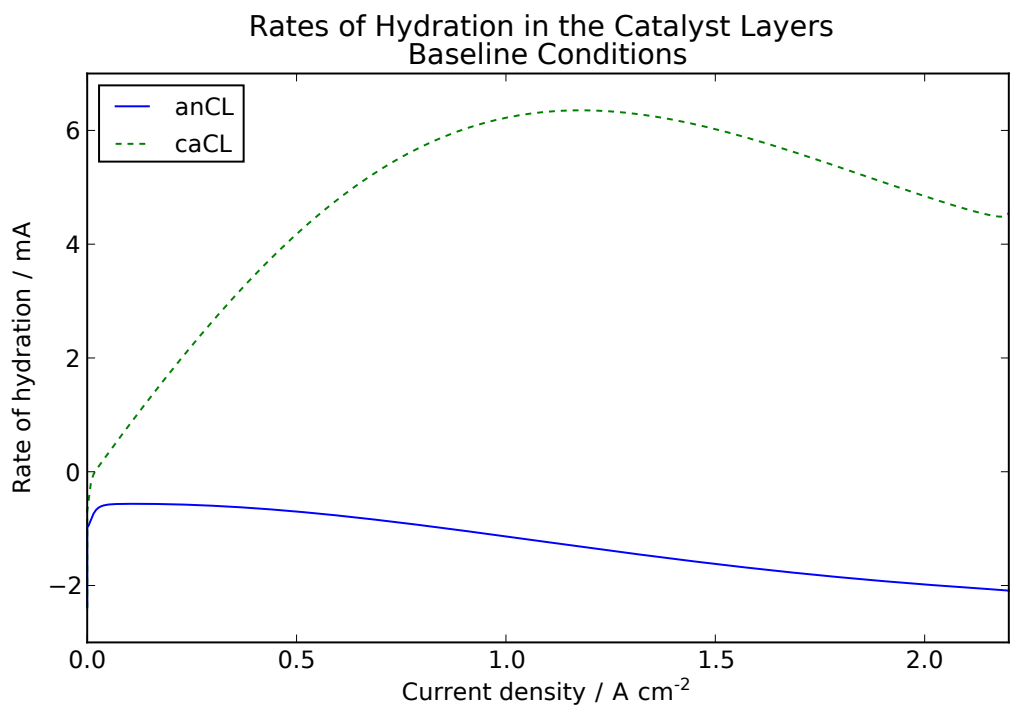


Figure 6.19: Rates of hydration during the baseline polarization.

6.2 Varying Temperature

6.2.1 Conditions

The temperature of the polarization test is changed to 40 and 80 °C from the baseline of 60 °C. As mentioned in Section 6.1.1, the selected temperature is applied to the reactant supplies, the exterior x-axis boundaries of the flow plates, and the initial conditions of the cell. All other settings are the same as for the baseline test (previous section).

6.2.2 Results and Discussion

The model statistics (numbers of both types of variables, number of states, and number of systems of equations) are the same as for the baseline test and the computation times are nearly the same. Unfortunately, the modeling tool requires that the model be re-translated when the temperature condition is changed. Sometimes this is necessary when a parameter has implications that may affect the structure of the underlying equations.

Figure 6.20 compares the polarization of the model against the experimental data. The same issues are apparent as in Section 6.1.2: the model over-predicts the activation region, under-predicts the Ohmic region, and does not show a gradual enough roll-off in the concentration region. The model does capture the trend towards higher performance at higher temperatures, but not to the extent of the experimental cell.

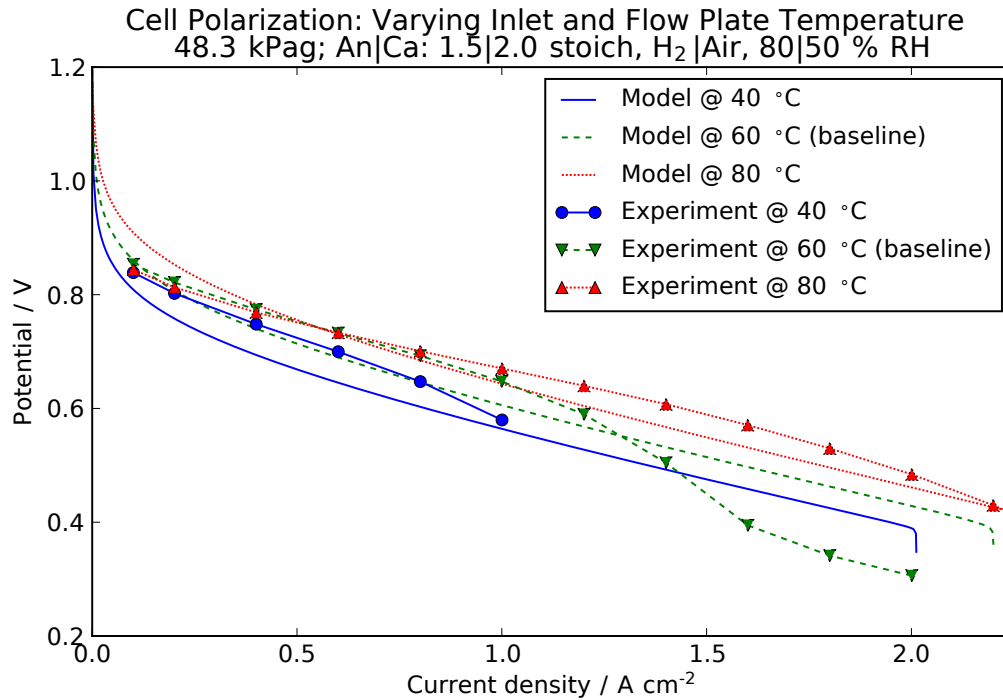


Figure 6.20: Polarization curves with varying temperature.

6.3 Varying Pressure

6.3.1 Conditions

The pressure of the polarization test is changed to 0 and 202.7 kPag from the baseline of 48.3 kPag. The selected pressure is applied to both of the outlets and to the initial conditions. All other settings are the same as for the baseline test (Section 6.1).

6.3.2 Results and Discussion

The model statistics are the same as for the baseline test and the computation times are nearly the same. Unfortunately, the modeling tool requires that the model be re-translated when the pressure is changed.

Figure 6.21 compares the polarization of the model against the experimental data. The same general issues exist as before. The model does capture the trend towards higher performance at higher pressure, but not to the extent of the experimental cell.

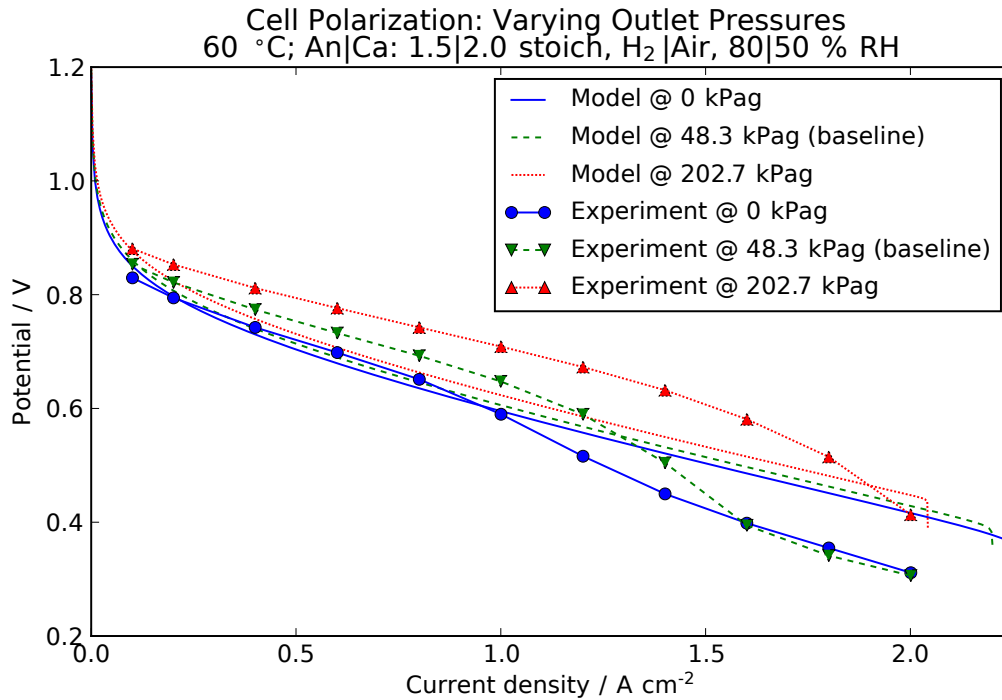


Figure 6.21: Polarization curves with varying outlet pressure.

6.4 Varying Anode Flow Rate

6.4.1 Conditions

The anode stoichiometric flow rate of the polarization test is changed to 1.1 and 2.0 from the baseline of 1.5. All other settings are the same as for the baseline test (Section 6.1).

6.4.2 Results and Discussion

The model statistics and translation time are the same as for the baseline test. The simulation time is only slightly different. It is possible to change the anode flow rate without re-translating the model.

Figure 6.22 compares the polarization of the model against the experimental data. Although the same general issues exist as noted in Section 6.2, the model and the experimental data both indicate that anode flow rate has little effect on the polarization. The supply of hydrogen generally does not limit the operation of the cell. The cathode supply does.

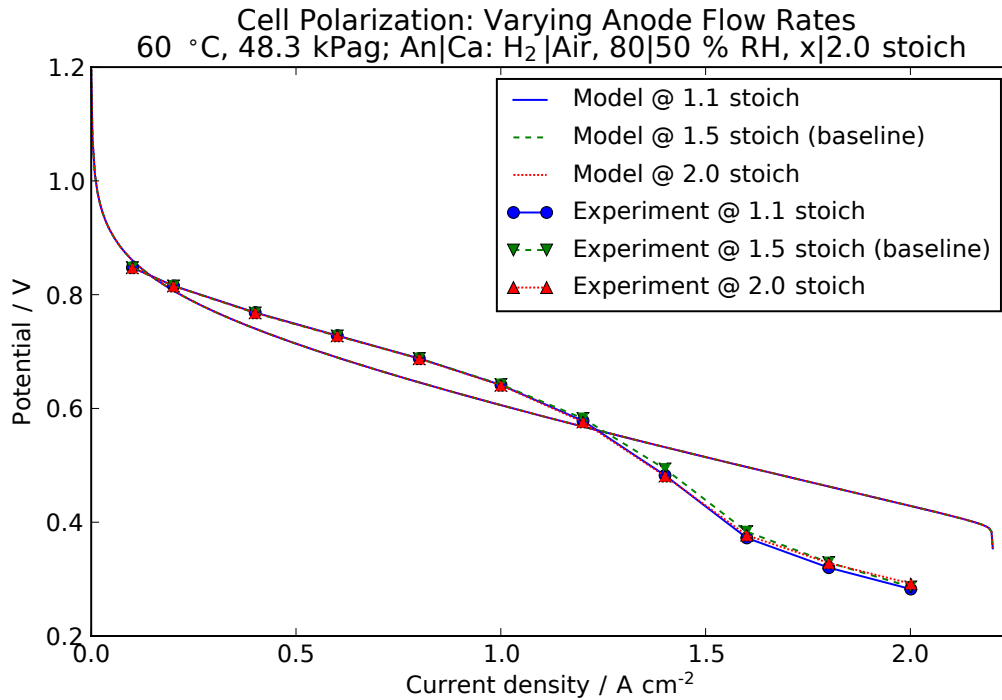


Figure 6.22: Polarization curves with varying anode flow rates.

6.5 Varying Cathode Flow Conditions

6.5.1 Conditions

The cathode stoichiometric flow rate of the polarization test is changed to 1.5 and 2.5 from the baseline of 2.0. In addition, the supply is changed to O₂ (no N₂) and the cell is operated at cathode stoichiometric flow rates of 7.177, 9.569, and 11.962. All other settings are the same as for the baseline test (Section 6.1).

6.5.2 Results and Discussion

The model statistics and translation time are the same as for the baseline test. The simulation time is nearly the same. It is possible to change the cathode flow rate without re-translating the model. However, the choice of air or O₂ requires re-translation.

Figure 6.23 compares the polarization of the model against the experimental data. In the case of air (Figure 6.23a), the effect shown by the model is much less than the effect in the experimental data. The model is affected very little in the Ohmic region, but the limiting current density changes considerably. In the case of O₂ (Figure 6.23b), the resistance should

decrease significantly in the Ohmic region, but the model does not show it. As was shown in Figures 6.3 and 6.4, the resistance is dominated by protonic transport through the PEM. The protonic resistance does not appear to change in the model. The only way for it to change is for (1) the mobility factors to change (they are currently fixed) or (2) the hydration of the PEM to change, affecting the degree of electro-osmotic drag.

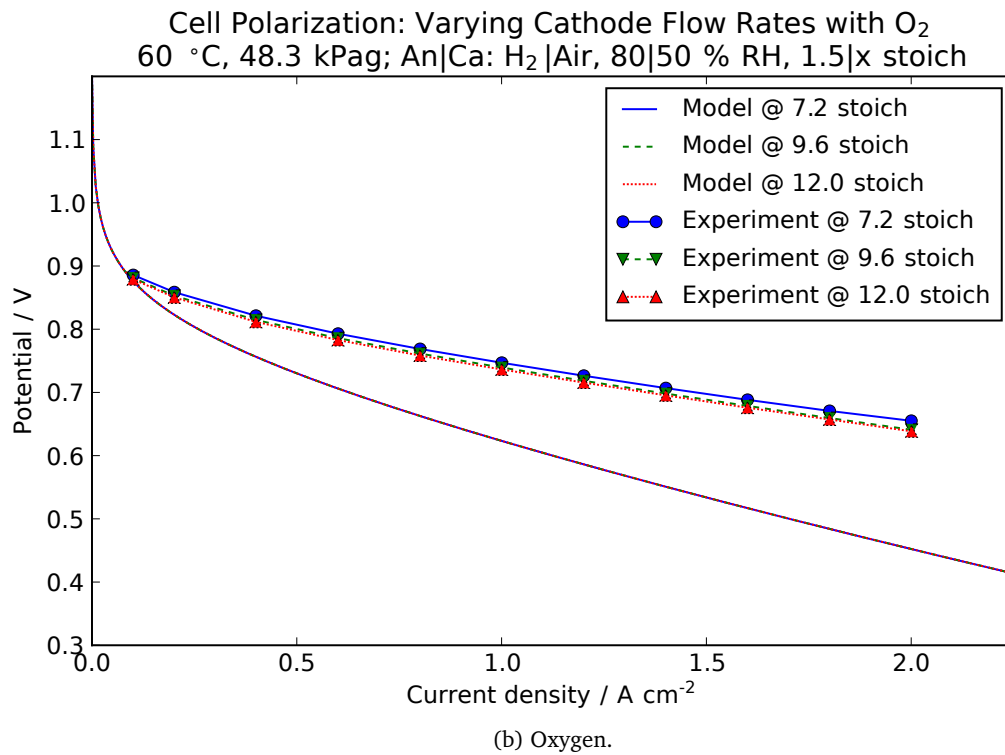
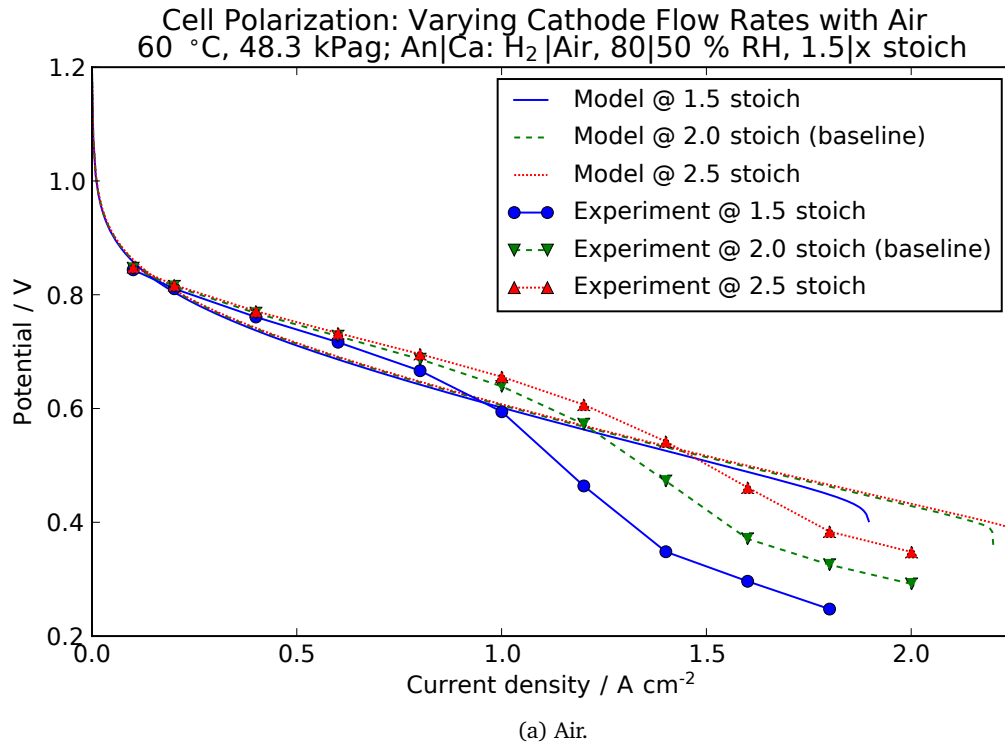


Figure 6.23: Polarization curves with varying cathode flow rates and compositions.

6.6 Varying Anode Humidity

6.6.1 Conditions

The relative humidity of the anode supply is changed to 60 and 100% from the baseline of 80%. All other settings are the same as for the baseline polarization test (Section 6.1).

6.6.2 Results and Discussion

The model statistics are the same as for the baseline test and the computation times are nearly the same. The modeling tool requires that the model be re-translated when the humidity of the anode supply is changed.

The anode humidity has little effect on the polarization, as shown in Figure 6.24 for both the model and the experimental cell.

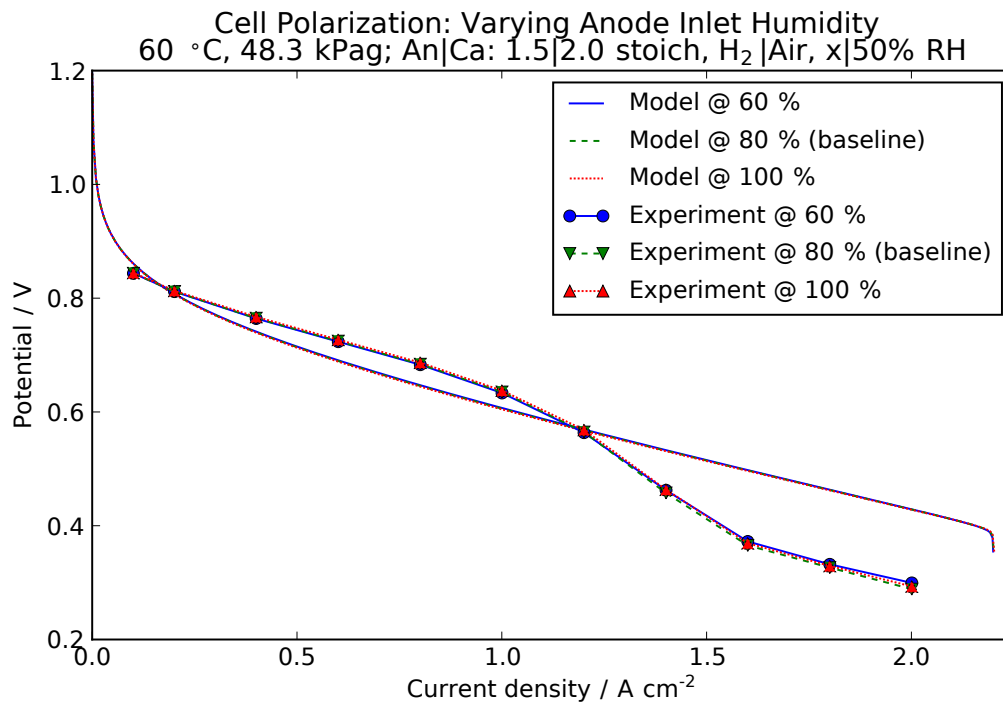


Figure 6.24: Polarization curves with varying humidity at the anode inlet.

6.7 Varying Cathode Humidity

6.7.1 Conditions

The relative humidity of the cathode supply is changed to 30 and 70% from the baseline of 50%. All other settings are the same as for the baseline polarization test (Section 6.1).

6.7.2 Results and Discussion

The model statistics and translation time are the same as for the baseline test and the simulation time is nearly the same. The humidity of the cathode supply can be changed without re-translating the model.

As shown in Figure 6.25, the cathode humidity has a slight effect on the polarization of the experimental cell in the concentration region. This reinforces the hypothesis that the rapid decrease in voltage from 1.0 to 1.5 A/cm² is due to the presence of liquid water. It may be the case that the water begins to fill the pores of the cathode GDL and then is released (possibly by a capillary effect or by the flow of vapor from the catalyst layer) above 1.5 A/cm². As was shown in Figure 6.16, liquid does fill the pores of the cathode GDL in the model. This increases the pressure gradient required for a given rate of O₂ transport. However, it seems that that only affects the limiting current density. It does not introduce a recoverable concentration roll-off.

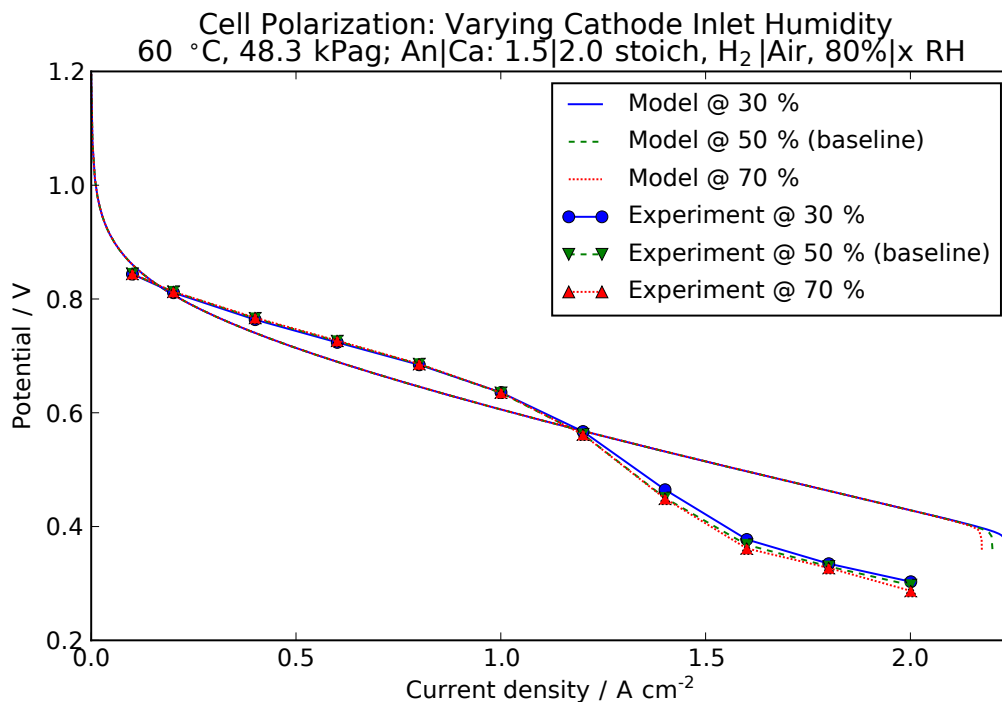


Figure 6.25: Polarization curves with varying humidity at the cathode inlet.

6.8 Simple Cell Model

A simplified fuel cell model is tested under the baseline conditions in order to establish a lower limit on the complexity of the model and the computation time.

6.8.1 Conditions

As discussed in Chapter 4, the simple cell model assumes the same temperature for all phases in each flow plate. This is in addition to the assumption in the standard cell model that all phases have the same temperature in the GDLs and catalyst layers. Liquid water is not included, and the ORR produces vapor instead of liquid. The GDL and catalyst layers are combined as shown in Figure 4.10c. In order to make the losses roughly the same as for the standard model, it was necessary to decrease the porosity of the combined layers to 0.16 (from 0.8). Also, the mobility factor between H^+ and H_2O was increased to 1 (from 0.02) and the protonic conductivity was increased to 0.8 S/cm (from 0.1 S/cm) in the combined layers (PEM unchanged).

The same base TestStand model (Figure 4.11) is used for this example, but its cell is replaced by the simple cell model.

6.8.2 Results and Discussion

Table 6.2: Modeling and simulation statistics for the simple fuel cell model.

	Standard model (with analysis)	Simple model	
		w/ analysis	w/o analysis
Number of variables	6887	4462	3564
Number of time-varying variables	2749	1650	866
Number of states	55	27	27
Sizes of the nonlinear systems of equations	None	None	None
Translation time	23 s	18 s	15 s
Simulation time	1.56 s	0.222 s	0.201 s

As listed in Table 6.2, the simplified cell model has 35% fewer total variables, 40% fewer time-varying variables, and roughly half of the number of states. It translates slightly more quickly than the standard model and simulates approximately seven times more quickly. The complexity of the model can be decreased further by disabling the analysis variables. Although

this significantly decreases the number of time-varying variables, it does not affect the number of states. Therefore, there is only a slight (9%) improvement in simulation time. The model could be simplified yet further by setting the thermal resistivities to zero, yielding an isothermal, lumped-capacitance cell.

Figure 6.26 compares the polarization of the simple cell model against the standard cell model. With the present parameter settings, the simple model has slightly less electrical resistance. The concentration loss appears earlier and more gradually, which is actually desirable based on the experimental results (Section 6.1.2).

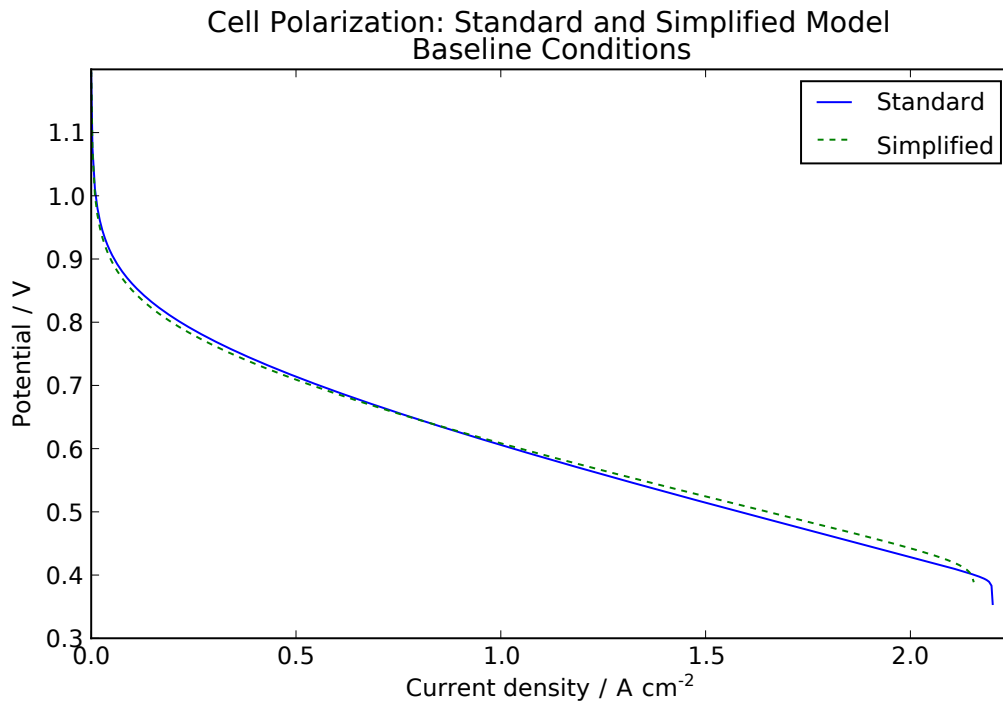


Figure 6.26: Polarization curves for the standard and simplified cell models.

Since liquid is not included, the relative humidity rises well above 100%, as shown in Figure 6.27. Another consequence of the exclusion of liquid is that the net hydration of the MEA is constant. Since the hydration process is modeled as phase change between the liquid and the ionomer (not between gas and ionomer) and liquid is not included, water cannot enter or exit the ionomer.

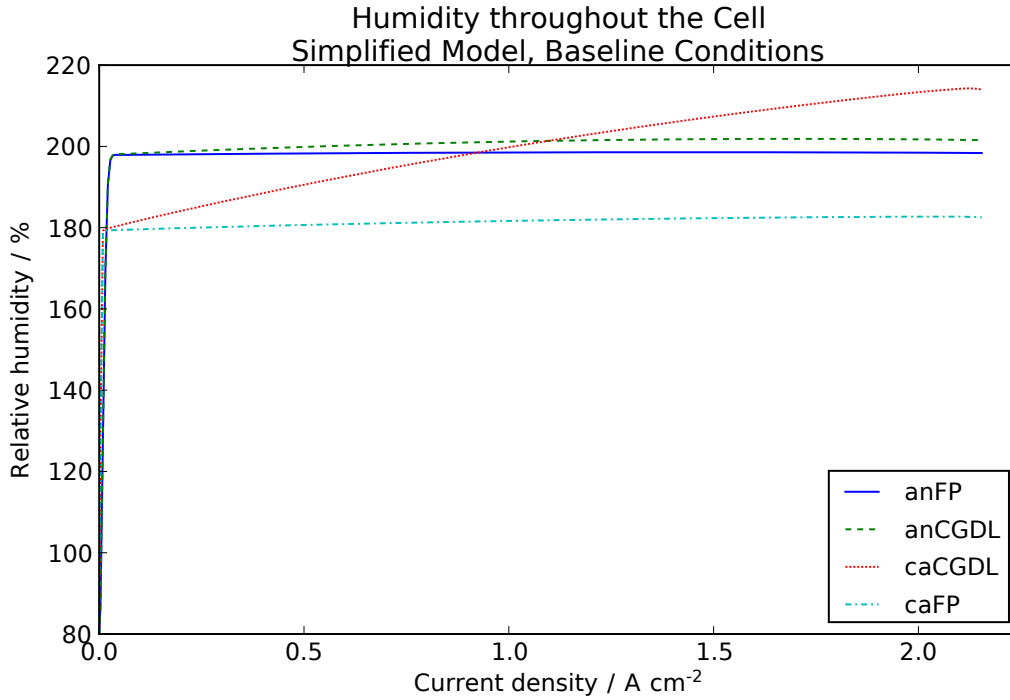


Figure 6.27: Relative humidity throughout the simplified cell model.

6.9 Segmented Cell

6.9.1 Conditions

In this section, a more complex configuration is considered by increasing the number of channel segments. A version of the standard fuel cell model is created with six subregions down the length of the channel (y direction). The electrical load is placed on the first segment, but the boundary temperature (60 °C) is applied to all of the segments. Liquid water is disabled. In order to make the losses without liquid roughly the same as for the standard model with liquid, it is necessary to change the porosity of the GDLs to 0.45 (from 0.8). The cell is operated under two conditions: the baseline conditions (Section 6.1.1) and the baseline conditions modified for fixed flow rates. The fixed flows are given by equivalent current densities of 2.4 and 3.2 A/cm² in the anode and cathode, respectively.

Table 6.3: Modeling and simulation statistics for the segmented fuel cell model.

	Stoichiometric flow		Fixed flow
	w/ analysis	w/o analysis	(w/ analysis)
Number of variables	31,980	24,835	31,990
Number of time-varying variables	12,711	6547	12,711
Number of states	266	266	266
Sizes of the nonlinear systems of equations	None	None	None
Translation time	67 s	48 s	69 s
Simulation time	18.8 s	18.1 s	11.3 s

6.9.2 Results and Discussion

As listed in Table 6.3, the segmented cell model has approximately 32,000 variables, roughly 40% of which are time-varying. There are approximately five times as many states as the standard, single-segment cell model. There would be six times as many if liquid water were included. As before, the analysis variables have little effect (4%) on the simulation time. The model runs 40% more quickly under fixed flow conditions.

Figure 6.28 shows the net polarization of the model under the two flow regimes. At 1.6 A/cm^2 , the supplies to the cell are equal under the stoichiometric and fixed flow conditions. Until that point, the cell performs better under the fixed flow condition. Shortly after that (at approximately 1.8 A/cm^2), the cell begins to reach its concentration limit in the fixed flow example.

Figure 6.29 shows the polarization of individual channel segments in the two operating regimes. Cell potential is on the independent axis since it is nearly equal among the segments. The upstream segments perform better than the downstream ones because the O_2 pressure is higher there. In the fixed flow example (Figure 6.29b), the last segment begins to provide less current as more current is drawn from the entire cell above 1.7 A/cm^2 . At that point, a positive feedback mechanism is established where an increase in net current requires more current from the upstream segments, further decreasing the O_2 available to the downstream segments. The last segment is the first to become completely depleted of O_2 . At that point, the simulation stops.

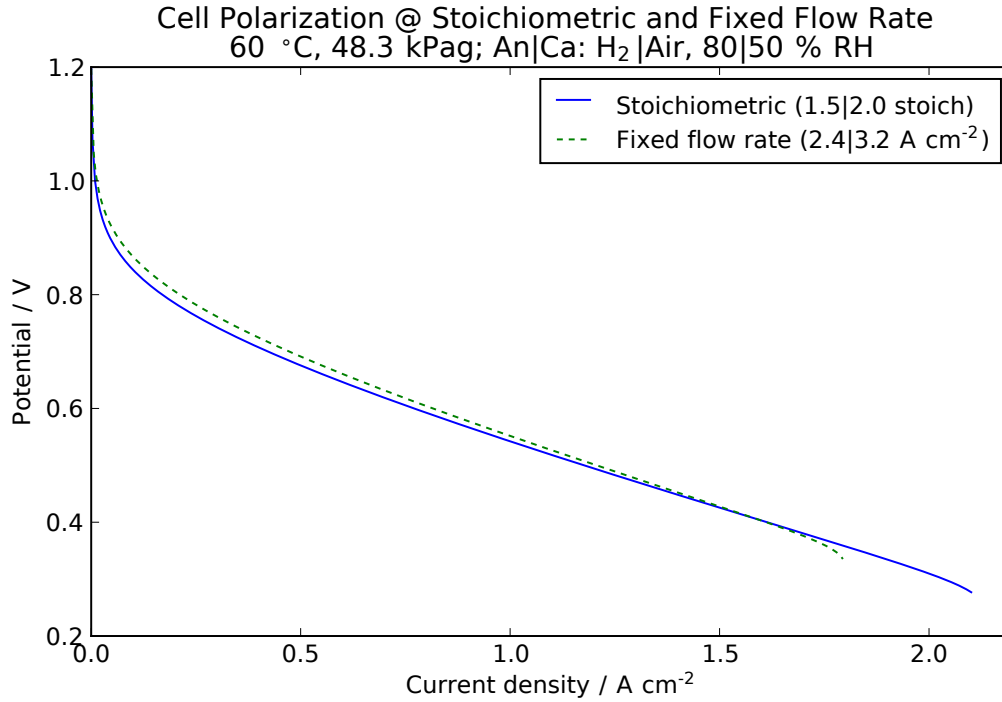


Figure 6.28: Net polarization of the segmented cell under stoichiometric and fixed flow.

Figure 6.30 shows the O₂ pressure in the cathode catalyst layer under the stoichiometric flow conditions. The pressure decreases fairly linearly with current density. The downstream segments have lower pressures. The pressure difference between the segments generally increases with current density because O₂ is consumed at a higher rate.

The upstream segments produce more electrical current due to the higher O₂ pressure and lower overpotentials, which are shown in Figure 6.32. Although the efficiency is higher in the upstream segments, those segments produce more heat due to the higher current. The rates of heat generation due to the ORR in the segments is shown in Figure 6.33. This causes the upstream segments to have higher temperature, as shown in Figure 6.31. The higher temperature further reduces the overpotential in the upstream segments due to the activation energy.

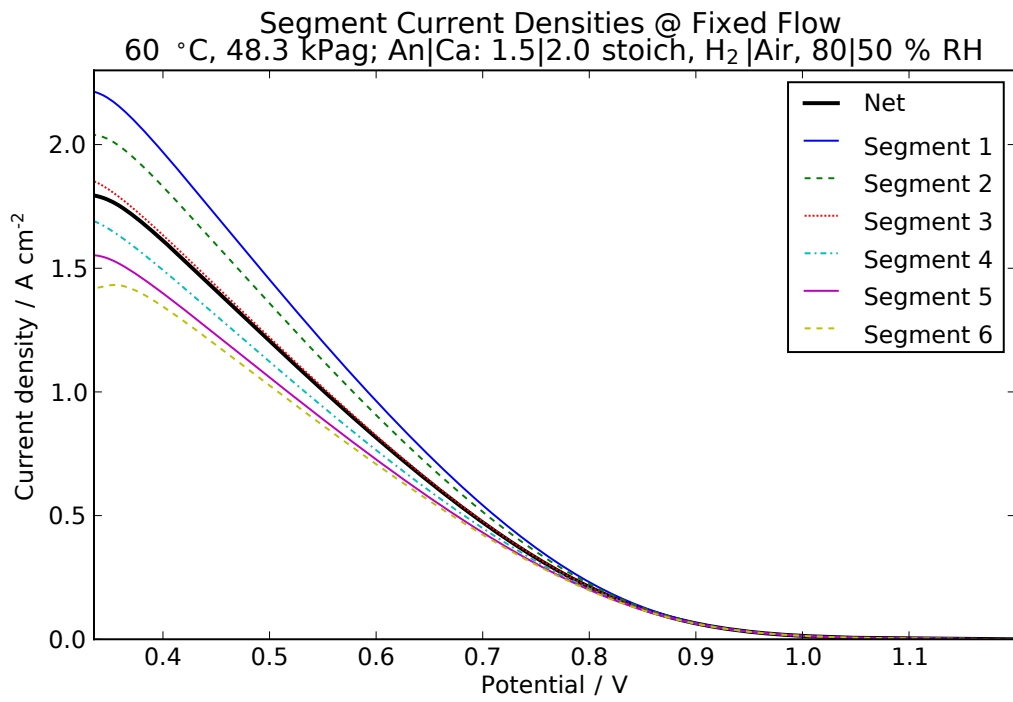
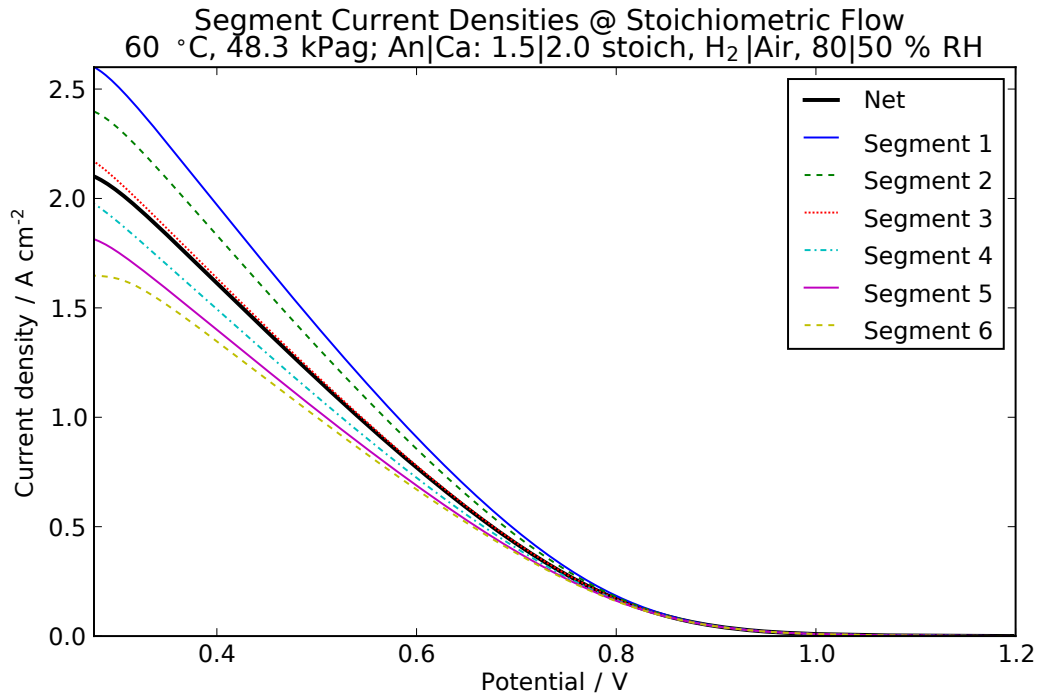


Figure 6.29: Pressure down the channels of the segmented cell.

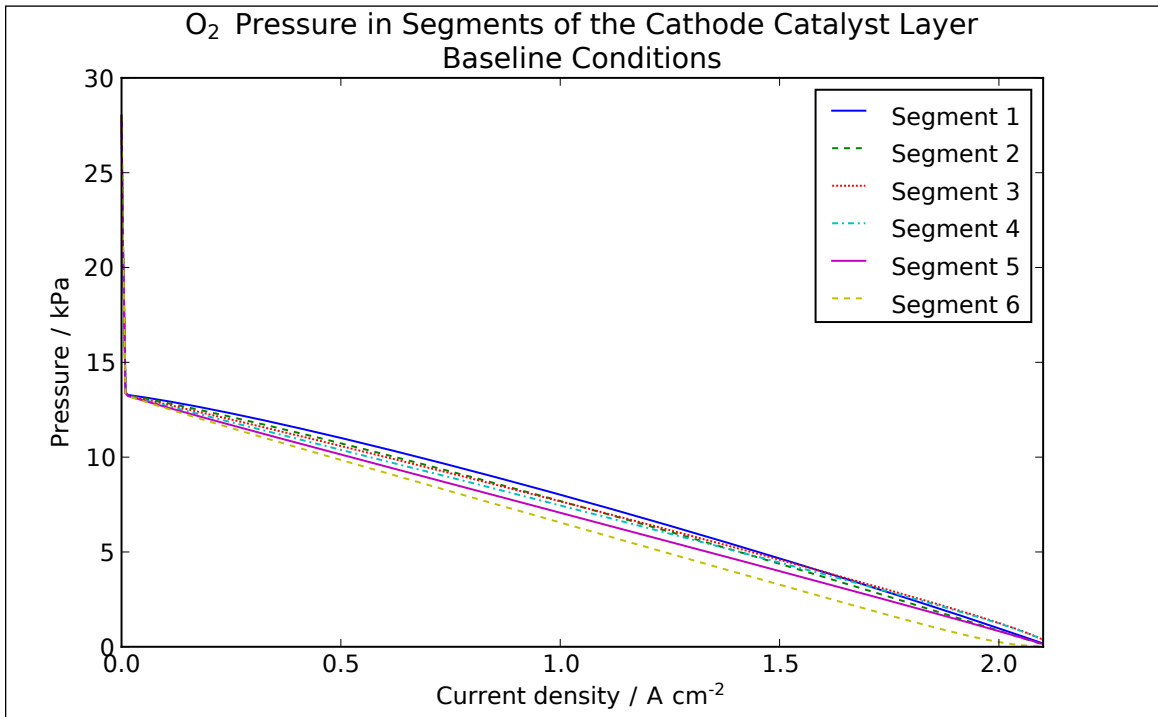


Figure 6.30: Pressure of O₂ in the cathode catalyst layer of the segmented cell.

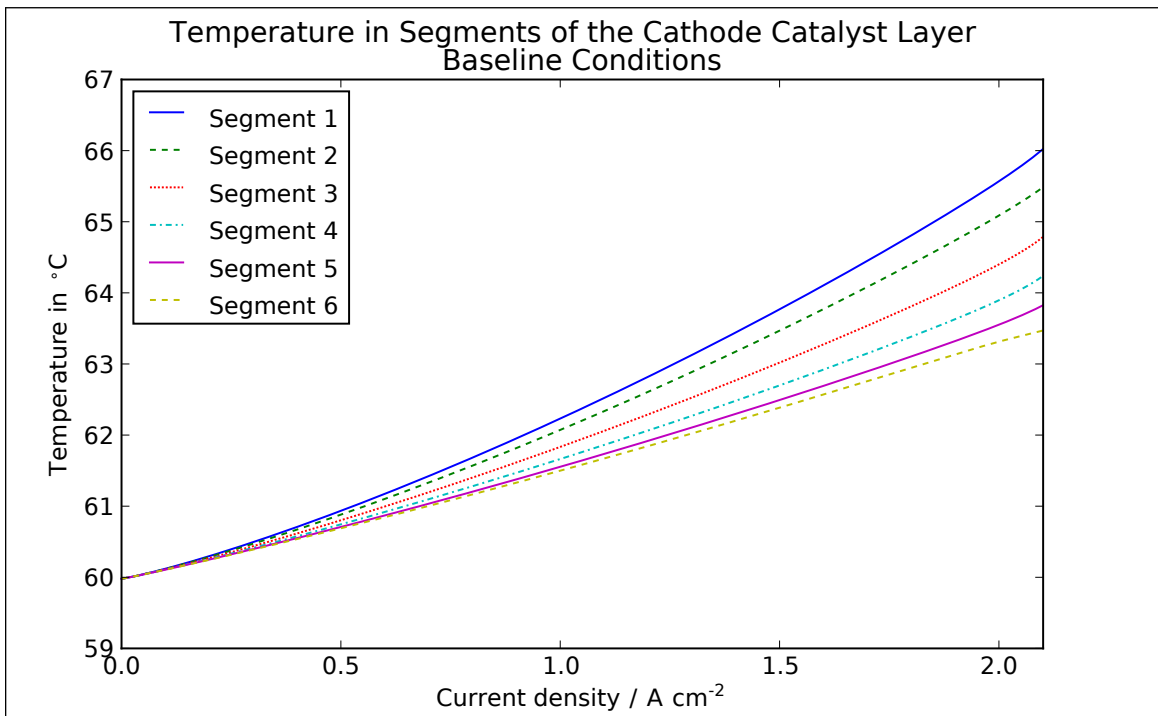


Figure 6.31: Temperature in the cathode catalyst layer of the segmented cell.

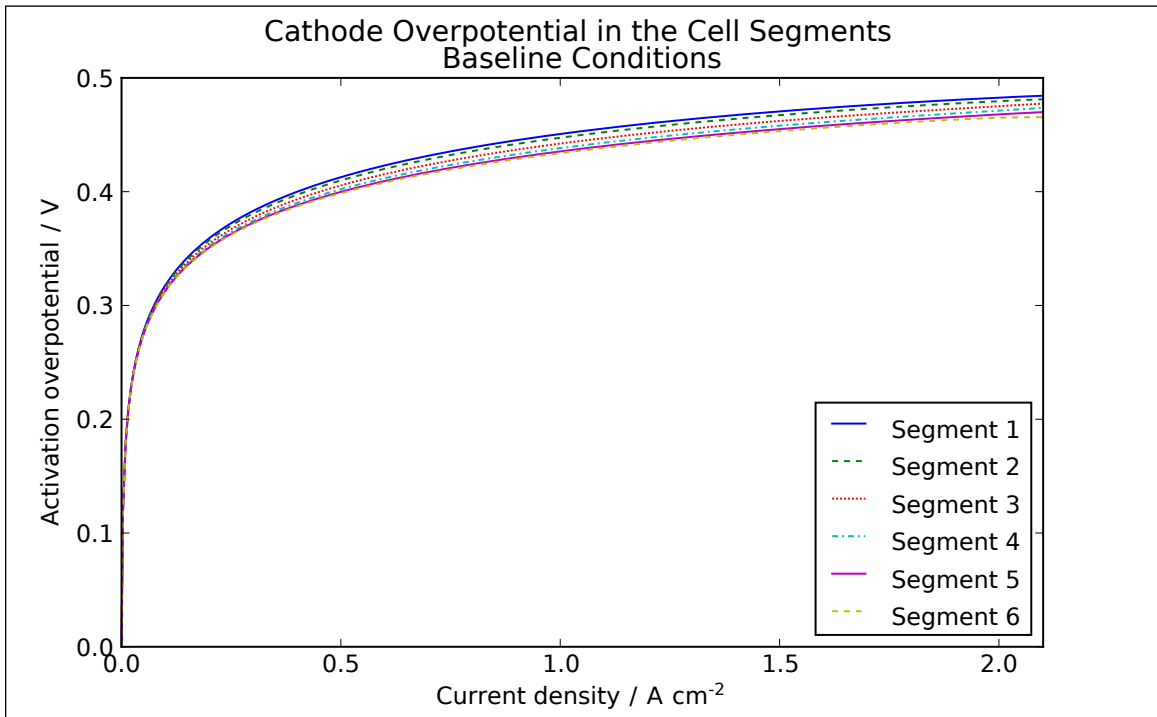


Figure 6.32: Overpotentials in the cathode of the cell segments.

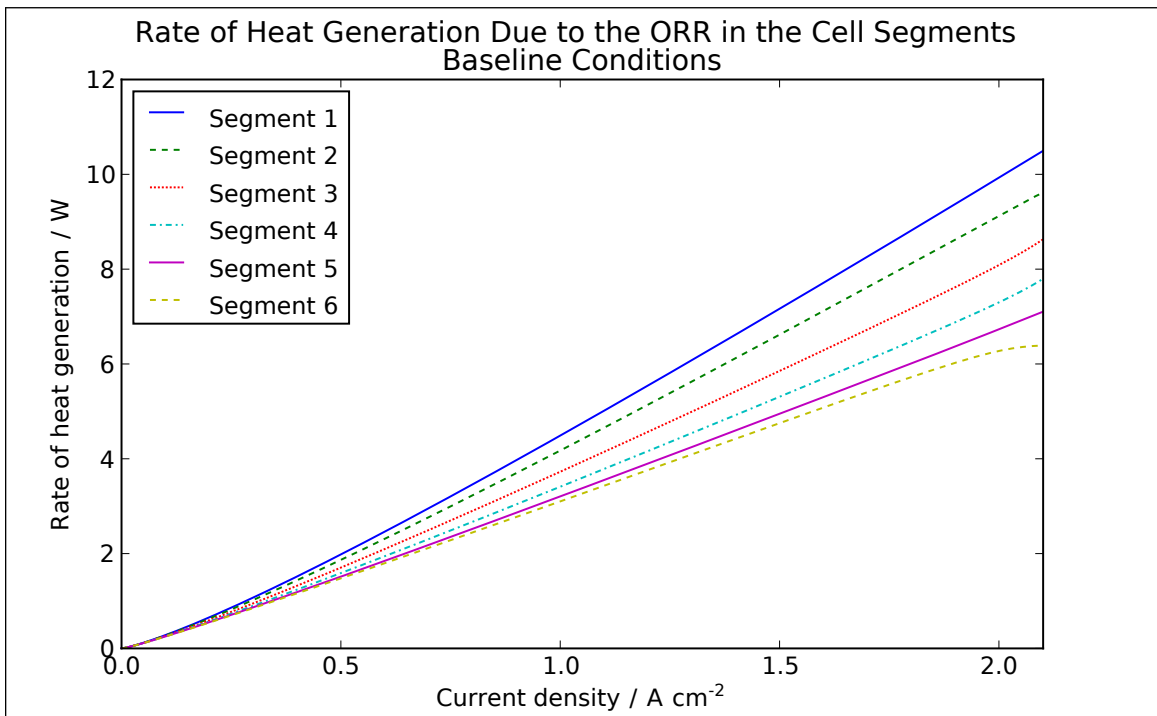


Figure 6.33: Rates of heat generation due to the ORR in the cell segments.

6.10 Cyclical Load

6.10.1 Conditions

This example demonstrates the dynamics of the cell model. The ramping current load of the TestStand model (Figure 4.11) is replaced with a sinusoidal current load as shown in Figure 6.34. The amplitude is 140 mA/cm^2 and the offset is 80 mA/cm^2 , as shown in Figure 6.35. To demonstrate the flexibility of the model, the minimum current density is negative (-60 mA/cm^2). The frequency is 0.3 Hz , and roughly three cycles are simulated. The anode and cathode supplies are fixed at equivalent currents of 300 and 400 mA/cm^2 . The electrolytic double layer models are included (not in the standard model).

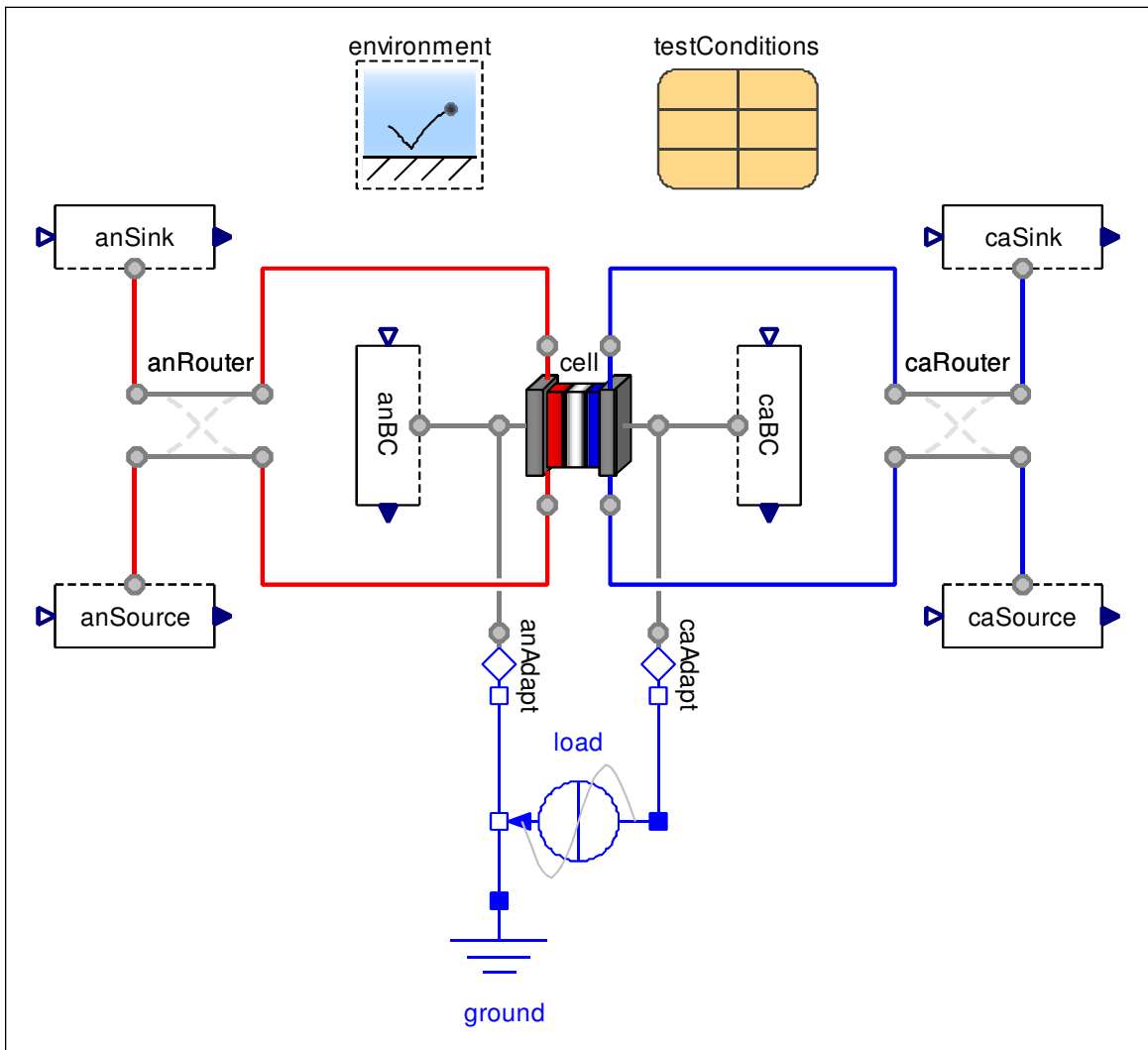


Figure 6.34: Model diagram for the test with a cyclical load.

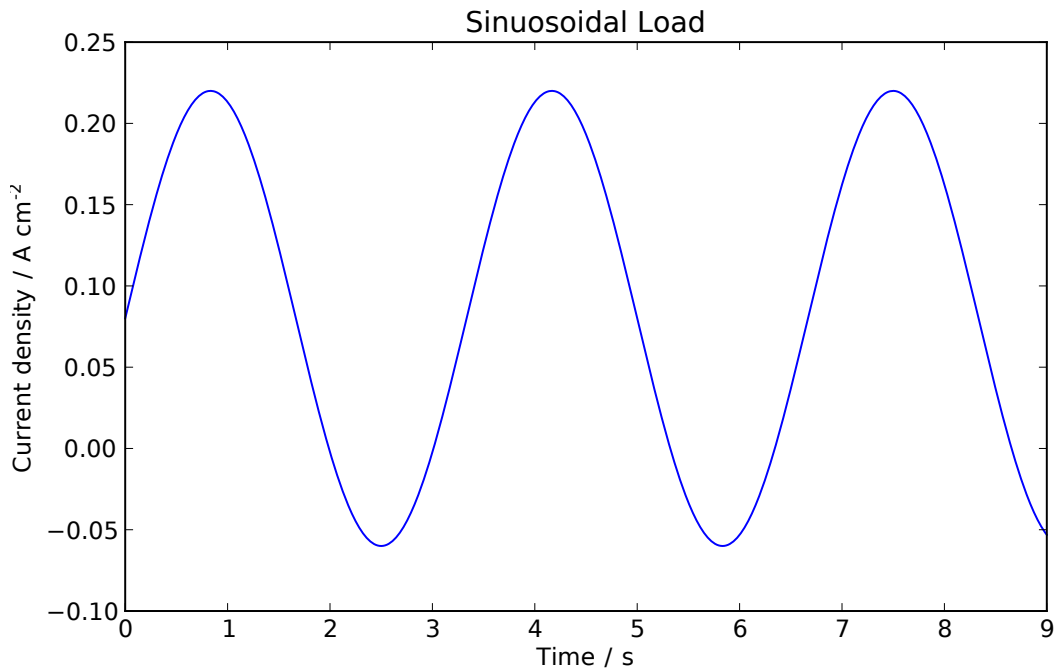


Figure 6.35: Cyclical load applied to the cell (sinusoidal, reversing).

6.10.2 Results and Discussion

Modeling and simulation statistics:

- Number of variables: 6942
- Number of time-varying variables: 2790
- Number of states: 57
- Sizes of the nonlinear systems of equations: None
- Sizes of the linear systems of equations: 1 set of 9, 1 set of 8, 1 set of 6, 2 sets of 5, 9 sets of 4, 20 sets of 3, 97 sets of 2
- Translation time: 23 s
- Simulation time: 4.03 s

As before, the model has no nonlinear systems of equations after translation. This example requires approximately twice as long to simulate as the baseline polarization test.

Figure 6.36 shows the current and voltage of the cell. The voltage lags the current primarily due to the double layer capacitance. There is a short settling period before a repeatable loop is established.

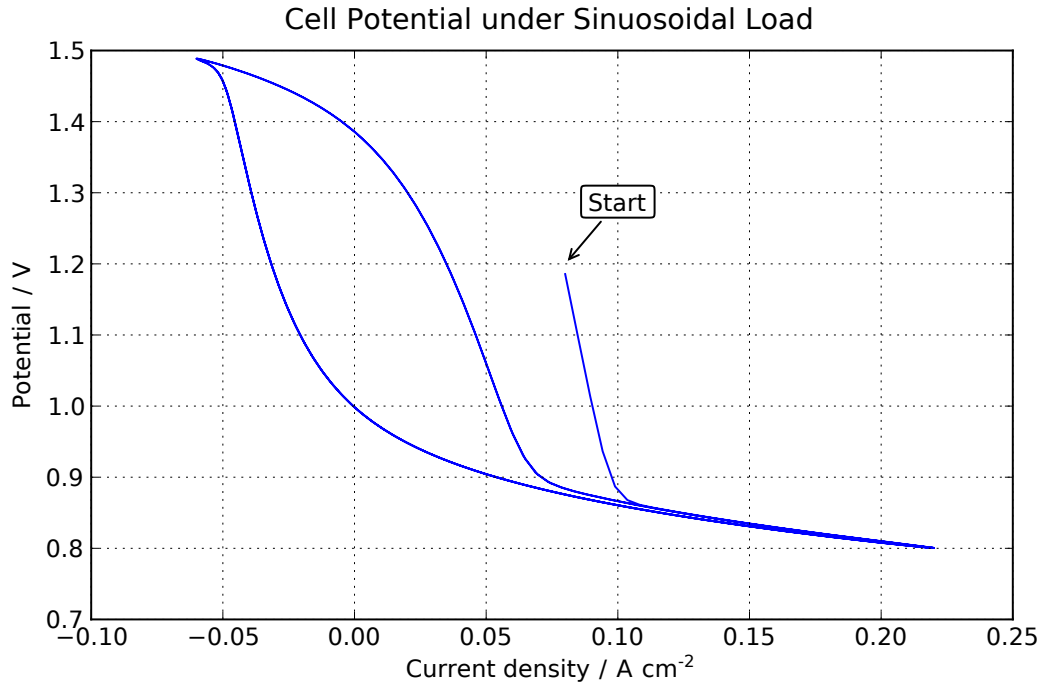


Figure 6.36: Current and voltage under cyclical load.

Figure 6.37 shows the temperatures throughout the cell. As before, the cathode catalyst layer reaches the highest temperature. However, when the current reverses, the reaction is endothermic and the temperature of the cathode GDL, cathode catalyst layer, and PEM decrease.

Figure 6.38 shows the pressure throughout the cell. The pressures are highest in the catalyst layer at the peak rate of production of H_2 and O_2 . The pressures are lowest at the peak rate of production of water, since the reactant gases must be drawn into the catalyst. The production of water also increases the relative humidity of the gas and fills the pores of the cathode catalyst layer and GDL, as shown in Figures 6.39 and 6.40.

Figure 6.41 shows the effect of the cyclical load on the hydration of the ionomer. There is little net change in the hydration of the MEA since the rate of phase change between the ionomer and the liquid is slow relative to the frequency of the load. However, when water is being produced, electro-osmotic drag causes the cathode side of the MEA to become more

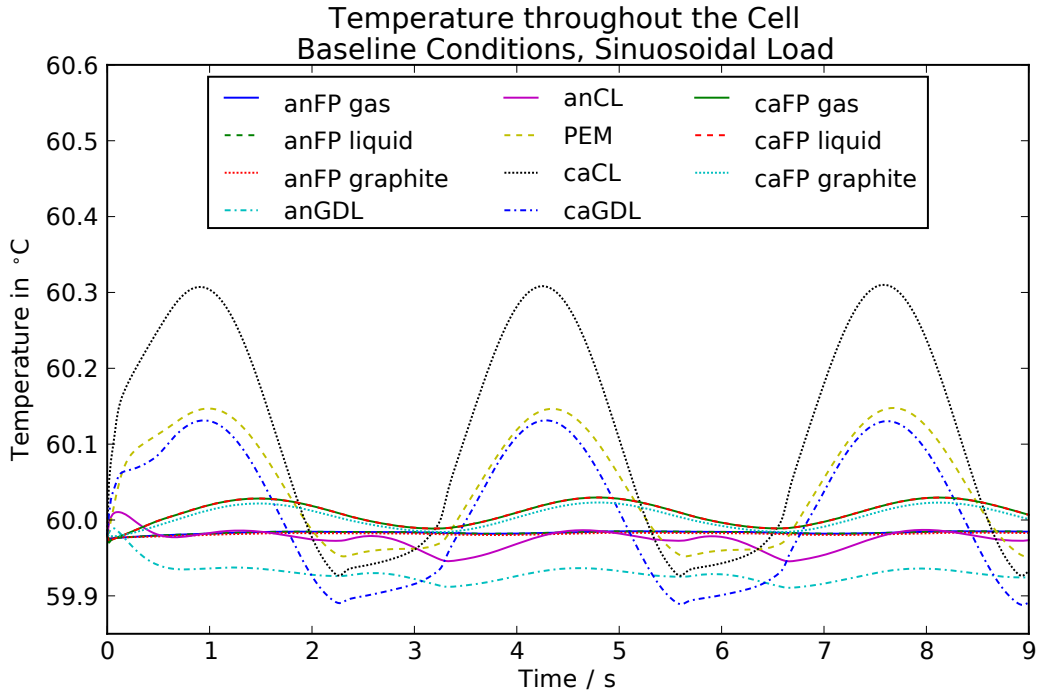


Figure 6.37: Temperatures of throughout the cell under cyclical load.

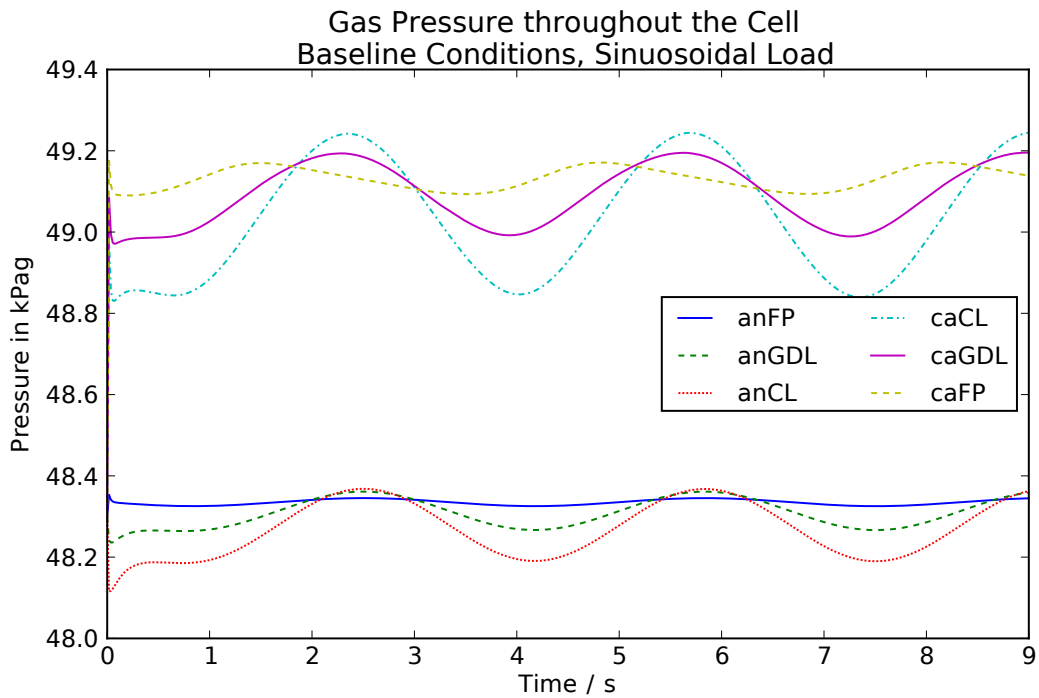


Figure 6.38: Gas pressure throughout the cell under cyclical load.

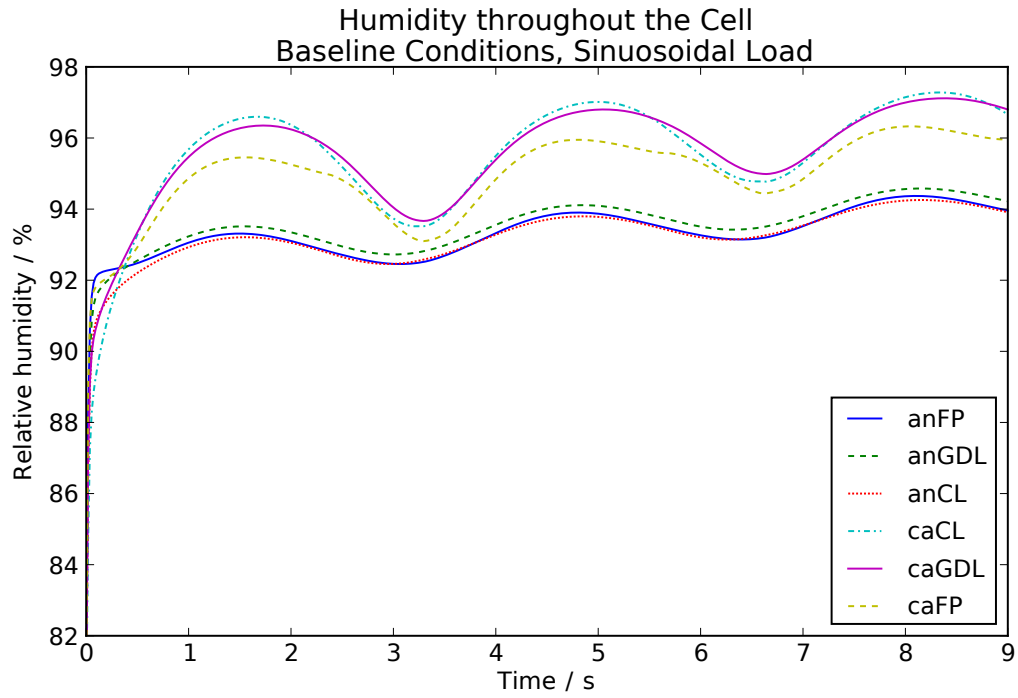


Figure 6.39: Relative humidity under cyclical load.

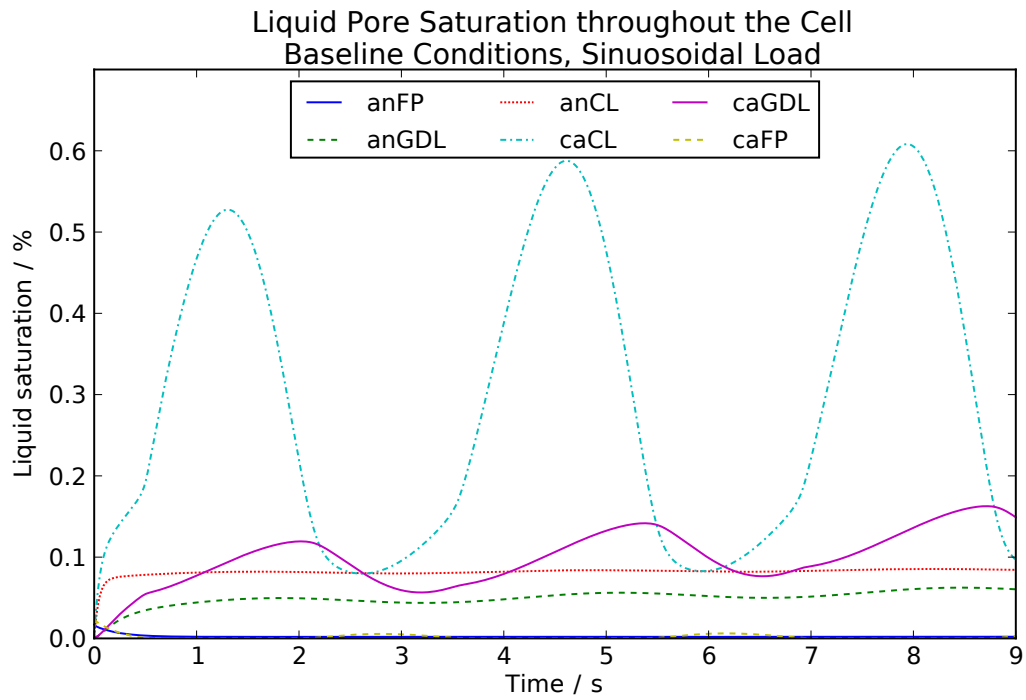


Figure 6.40: Liquid pore saturation under cyclical load.

hydrated at the expense of the anode side. When water is consumed, protons travel to the anode while dragging water and hydrating the anode side. On average, the cathode side is more hydrated because the sinusoidal offset is positive—biased towards water production in the cathode.

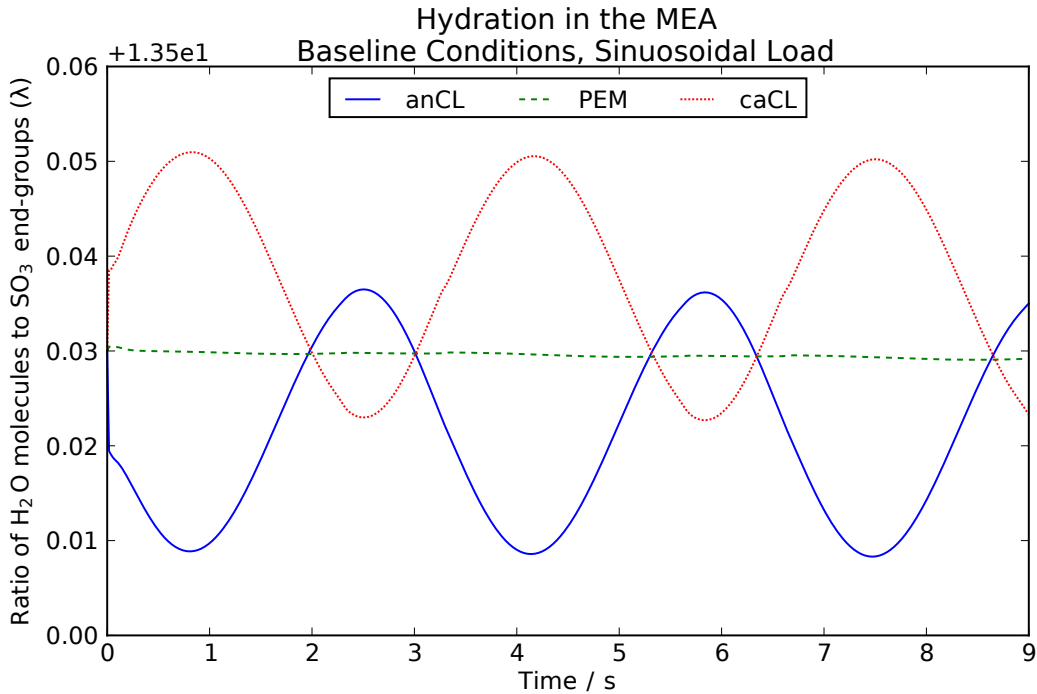


Figure 6.41: Hydration under cyclical load.

6.11 Summary

This chapter provided examples of the fuel cell model. Polarization tests were performed and compared to experimental results. The qualitative trends are in basic agreement with the experimental results. However, there are significant quantitative differences which may be due to hydrogen crossover and more complex behavior of liquid water in the experimental cell. The differences could also be due to the large number of parameters in the model and the remaining uncertainty in their settings. As demonstrated by the examples of simplified and segmented cell models, the modeling framework is reconfigurable. As was shown under a cyclical load, the fuel cell model is also dynamic.

CONCLUSIONS

7.1 *Recapitulation*

In the first chapter, the advantages of declarative modeling were presented. Although declarative modeling has been successfully applied to many domains (e.g., electrical, thermal, and mechanical), there is no widely accepted way to apply it to chemical/fluid devices. One of the main reasons is that these devices involve both advection and diffusion. It is sometimes appropriate to assume pure diffusion or pure advection, but each assumption leads to different implementations in declarative language. The goal of this work has been to realize the advantages of declarative modeling for complex physical systems that involve both advection and diffusion to varying degrees in multiple domains.

















The first research question established in the first chapter was the following:

RQ1: How can we create a generic declarative framework to model systems with processes that exhibit coupled advection and diffusion?

The approach that is presented, justified, and demonstrated in this work is to use effort/flow pairs with a custom, generic upstream discretization scheme. It reduces to the central difference scheme under pure diffusion and the upstream scheme under pure advection. The transition between these two extremes is gradual, and no switching events are generated. No mathematical causality is assigned. No nonlinear systems of equations are generated, regardless of the size of the physical system. The approach is compatible with traditional declarative connectors because there is exactly one effort for each flow. However, unique choices are made regarding the quantities used as efforts and flows. The effort/flow pairs established in Table 7.1 allow exact conservation equations for material, translational momentum, and energy.ⁱ

ⁱThe only caveat is that the thermodynamic pressure at a boundary is biased by the dynamic pressure unless nonlinear equations are introduced. However, this is not a violation of the conservation of momentum or energy since the error is consistent between neighboring subregions.

Table 7.1: Effort/flow pairs of the connectors (duplicate of Table 4.1).

Name	Effort	Flow	Within icon(s)
Material	Pressure p [$\text{ML}^{-1}\text{T}^{-2}$]	Current \dot{N} [NT^{-1}]	 
Translational	Velocity ϕ [LT^{-1}]	Force $\dot{m}\phi$ [LMT^{-2}]	    
Thermal diffusive	Temperature T [$\text{L}^2\text{MN}^{-1}\text{T}^{-2}$]	Heat flow rate \dot{Q} [L^2MT^{-3}]	   
Thermal advective	Temperature times specific entropy Ts [$\text{L}^2\text{MN}^{-1}\text{T}^{-2}$]	Heat flow rate \dot{Q} [L^2MT^{-3}]	
Amagat	Pressure p [$\text{ML}^{-1}\text{T}^{-2}$]	Partial volume V [L^3]	
Dalton	Volume V [L^3]	Partial pressure p [$\text{ML}^{-1}\text{T}^{-2}$]	
Chemical	Chemical potential g [$\text{L}^2\text{MN}^{-1}\text{T}^{-2}$]	Current \dot{N} [NT^{-1}]	
Stoichiometric	Rate of reaction \dot{N} [NT^{-1}]	Net chemical potential g [$\text{L}^2\text{MN}^{-1}\text{T}^{-2}$]	

The second research question was:

RQ2: How can the equations be best implemented to reflect the physical structure of a device and support reconfiguration?

By implementing individual chemical species as models with connectors, the approach follows the advice of Cellier to reduce the semantic distance between the lowest graphical objects and the textual equations [16]. The *Species* model describes both storage and transport, which is more physically representative than the traditional lumped parameter method (i.e., modeling a fluid network as a set of alternating volumes and pipes). The model equations are implemented in an equation-based, object-oriented (EEO) modeling language. The object-oriented structure corresponds to a physical device: models of chemical species are instantiated within phases and phases are instantiated within subregions. Multiple subregions are combined to form regions, and regions are connected to build assemblies such as a fuel cell.

Material and energy are stored within the subregions. Momentum is stored at the boundaries. This is essentially the staggered grid concept, which is known to avoid the possibility of a wavy pressure and velocity profile [51].

The EEO approach is even used for chemical reactions, where the stoichiometry of a reaction is represented neatly by submodels for each species (Figure 4.3). It is also used to constrain the relations between the pressures and volumes of species and phases (e.g., Amagat's law, Dalton's law, and the Young-Laplace equation).

The implementation is highly reconfigurable, as indicated by the wide array of examples in Chapter 5 that are all built by assembling the same subregion in various ways. Species can be included or excluded. Assumptions can be applied using parameters. It is possible to directly couple certain properties (e.g., temperature) within sets of species. Through index reduction, this reduces the number of states and yields a simpler model.

The third research question had three parts:

RQ3: How appropriate is the framework for modeling all the relevant physical phenomena of an electrochemical device such as a fuel cell?

- a:** For which processes is it necessary to model mixed advection and diffusion? Where is it appropriate to assume pure advection or pure diffusion?
- b:** What characteristics do the models exhibit that would not be present given an imperative formalism?
- c:** Which combinations of accuracy and speed can be achieved by adjusting fidelity?

A fuel cell model was successfully demonstrated using the modeling framework. The model simulates very efficiently and offers a high level of detail regarding physical properties and interactions, as shown in Chapter 6. However, the physical accuracy is currently less than desirable. It appears that this could be due to parameter settings, but it will be difficult to know for sure until the fuel cell model library is thoroughly vetted. For this reason and in the spirit of open-source collaboration, the library has been made publicly available.

Regarding the first subquestion (RQ3a), advection and diffusion are generally both considered for transport processes. Often advection or diffusion dominates, but using the modeling framework, it is convenient and does not appear to introduce unmanageable complexity to always consider the possibility of both advection and diffusion. The exception is solid materials. For solids, the base `Species` model is extended to create a simplified version that does not include material transport. Thus, advective transport is eliminated. For exchange (as opposed to transport), it has been appropriate to consider that advection and diffusion occur along separate pathways. The general advective/diffusive equation reduces to special cases for pure advection and diffusion. Pure diffusion is assumed for interactions that do not involve chemical reactions or phase change, as there is no material exchange. Pure advection is assumed for the exchange of momentum and energy among species along with chemical reactions and phase change.

Regarding the second subquestion (RQ3b), declarative modeling has been beneficial for several reasons which have been demonstrated. The first is index reduction, which has been used to create versions of the fuel cell model which do and do not assume the same temperature of all phases in the flow plates. Since index reduction is performed automatically, these types of assumptions are easy to manage. The second reason is model reuse. It would not have been

possible to demonstrate all of the examples in Chapter 5, given all of the various boundary conditions, with a single base imperative model. Third, the fuel cell model simply could not have been created in an imperative, object-oriented formalism given the level of complexity of the interactions that were included. Djilali states that “[o]ne of the most challenging aspects of computational modelling of PEMFCs is the multi-physics nature of the transport processes, and the coupling between these processes [...]” [74] EOO modeling has helped with that challenge. The fourth reason is computational efficiency. The algebraic manipulation necessary to create a numerically efficient segmented fuel cell model with 42 subregions (7 layers × 6 segments) and 12,711 time-varying equations could not be completed manually. Yet the declarative modeling tool performs this task in 67 s on a current laptop computer. The compiled model takes less than 19 s to simulate. Finally, the EOO implementation has allowed the model to be structured like the physical fuel cell. This is an advantage because as stated by Franke et al., “[t]he understanding of simulation models is generally simplified if the modular model structure corresponds to the structure of actual physical devices.” [53]

Regarding the third subquestion (RQ3c), a wide fairly range of fidelity has been demonstrated, especially in the fuel cell model. The standard fuel cell model yields a polarization curve in 1.56 s with 55 states, whereas a simplified model simulates in 0.22 s with 27 states. A segmented cell with six segments down the length of the channel has 266 states and simulates in 18.8 s. The effect of model detail on physical accuracy is currently difficult to establish since there is still a significant difference between the results of the model and experiment.

The fuel cell model fits in the space of physical detail and computational complexity somewhere between simple zero-dimensional (0D) models and computational fluid dynamics (CFD) models with high spatial resolution. This is an important gap because 0D models often do not provide enough insight into physical behavior, yet CFD models are unwieldy for dynamic design studies. The models simulate quickly enough to be manageable for design studies of the fuel cell or combined with other models (e.g., *Modelica.Fluid*) to study larger systems. The simplified cell model may be simple enough to run in real time, although the fast simulation time may be largely due to the variable-step solver, which is not available in real-time, embedded simulation.

7.2 Contributions

7.2.1 Primary

- Creation of an upstream discretization scheme that is suitable for declarative implementation
 - Avoids the numerical singularity of the exponential scheme [51] at pure diffusion
 - Avoids the switching behavior of the upwind scheme upon flow reversal
 - Used in the fuel cell model library and a one-dimensional (1D) fluid network to represent a Rankine power plant:
 - BINDER, W. R., PAREDIS, C. J. J., and GARCIA, H. E., “Hybrid energy system modeling in Modelica,” in *Proceedings of the 10th International Modelica Conference*, (Lund, Sweden), Modelica Association, Mar. 2014 (accepted)
- Creation of a set of dynamic chemical/fluid/thermal equations that are well-structured for implementation in EOO language
 - Includes the transport, storage, and exchange of material, momentum, and energy
 - Exact conservation equations
 - No nonlinear systems of equations
 - No switching equations except for chemical reactions and phase change
 - Avoids the problems of the dusty gas model noted in [197, 198]
- Development of derivations that relate the model equations to selected theories in solid state physics, fluid dynamics, mass and heat transfer, electrochemistry, and thermodynamics (Chapter 3 and Appendix A)
- Derivation and implementation of equations that relate various thermodynamic properties to (1) specific heat capacities represented as functions of temperature in the form of McBride et al. [142] and (2) a virial equation of state where the coefficients are polynomials of temperature (Sections 3.2 and B.3)
 - Suitable for incompressible species and ideal gases, but more general than either
 - Properties are exact given the virial coefficients and heat capacity coefficients
 - Polynomials are supported up to any degree

- Creation of FCSys, an open-source Modelica library for modeling fuel cells (Chapter 4 and Appendix B)
 - The first three-dimensional (3D) fluid/thermal/chemical model in Modelica
 - The first declarative, physics-based fuel cell model
 - Key features:
 - Multi-component, multi-phase
 - Based on first principles
 - Numerically efficient
 - Modular and reconfigurable in terms of spatial resolution and dimensionality, choices of species, and assumptions about properties and processes
 - Included phenomena:
 - Diffusion and pressure-driven transport
 - Dynamic storage of material, momentum, and energy
 - Electrochemical reactions
 - Phase change (of H₂O among three phases: gas, liquid, and absorbed in ionomer)
 - Binary diffusion
 - Electrical conduction
 - Thermal conduction and convection
 - Electro-osmotic drag
 - Capillary pressure
 - Meets the requirements of a good, general-purpose, object-oriented thermo-fluid modeling framework established by Franke et al. [53]
 - Available for download from <http://kdavies4.github.io/FCSys/> and the Modelica web page at <https://www.modelica.org/libraries>
 - Published in the following papers:
 - DAVIES, K. L., PAREDIS, C. J., and HAYNES, C. L., “Library for first-principle models of proton exchange membrane fuel cells in Modelica,” in *Proceedings of the 9th*

International Modelica Conference, (Munich, Germany), Modelica Association, Sep. 2012

- DAVIES, K. L., MOORE, R. M., and BENDER, G., “Model library of polymer electrolyte membrane fuel cells for system hardware and control design,” in *Proceedings of the 7th International Modelica Conference* (CASELLA, F, ed.), (Como, Italy), Modelica Association, Linköping University Electronic Press, Sep. 2009
- DAVIES, K. L., HAYNES, C. L., and PAREDIS, C. J., “Modeling reaction and diffusion processes of fuel cells within Modelica,” in *Proceedings of the 7th International Modelica Conference* (CASELLA, F, ed.), (Como, Italy), Modelica Association, Linköping University Electronic Press, Sep. 2009
- DAVIES, K. L. and MOORE, R. M., “Object-oriented fuel cell model library,” *Electrochemical Society Transactions*, vol. 11, pp. 797–808, Oct. 2007

- Surveys of literature related to:
 - EOO modeling languages (Section 2.1)
 - Approaches to modeling fluid systems in Modelica (Section 2.2)
 - Fuel cell models, with an emphasis on declarative models (Section 2.3)
- Model-based investigation of a proton exchange membrane fuel cell (PEMFC) including losses and heat generation, water transport and storage, and electro-osmotic drag (Section 6.1)
- Assessment of the accuracy of the PEMFC model’s polarization curves under various temperatures, pressures, humidities, and flow rates (Sections 6.1–6.7)
- Assessment of the computational complexity of the fuel cell models (Chapter 6)
- Creation of a flexible method to establish systems of physical units from fundamental constants (Sections 4.3 and B.72)
 - Key features:
 - Supports unit conversion inherently
 - Relates all units, including those of SI (besides the lumen), to six fundamental physical constants

- No other empirical constants are required
- Organizes quantities by physical dimensionality
- Initial implementation published as follows:
 - DAVIES, K. L. and PAREDIS, C. J., “Natural unit representation in Modelica,” in *Proceedings of the 9th International Modelica Conference*, (Munich, Germany), Modelica Association, Sep. 2012

7.2.2 Secondary

- Creation of ModelicaRes, an open-source Python package to analyze and plot the results of Modelica simulations
 - Used to analyze data and create all the plots for Chapters 5 and 6
 - Available online at <http://kdavies4.github.io/ModelicaRes/> and from the Modelica web page at <https://www.modelica.org/tools>
- Based on the initial work of Yannick Chopin, the development of a module for Sankey diagrams in matplotlib
 - Used to create Figures 6.2 and 6.12
 - Now a standard part of matplotlib (documentation at http://matplotlib.org/api/sankey_api.html)
- Submission of several functions for the Modelica Standard Library
 - Used in the fuel cell model library
 - Included in the `Modelica.Math.BooleanVectors` package
- Translation of a solid oxide fuel cell (SOFC) model from C code to Embedded MATLAB for real-time simulation
 - Led to the following publication:
 - HUGHES, D., FORD, J. C., DAVIES, K. L., HAYNES, C. L., WEPFER, W., and TUCKER, D., “A real-time spatial SOFC model for hardware-based simulation of hybrid systems,” in *Proceedings of the 9th Fuel Cell Science, Engineering and Technology Conference*, (Washington, DC), ASME, Aug. 2011

7.3 Future Work

- Re-derive the physical topics and laws (Table 3.1) that relate to material transport. As implemented, the momentum balances are located at and are normal to the boundaries of a subregion. Material advection and diffusion are included within the subregion and are not distinguished. This should be proven to relate directly to the established theories.
- Further calibrate and validate the fuel cell model. In Chapter 6, the free parameters were manually calibrated. An accurate calibration would require a sensitivity analysis and parameter identification. The calibration and validation should consider dynamic behavior and the current distribution of the segmented cell model. As stated by Meng and Wang [90], “an experimental validation of a PEFC model based on the polarization curve alone is insufficient, and [...] detailed current density distribution data in the along-channel direction is essential.”
- Further investigations using the fuel cell model:
 - Run the segmented cell under counter flow.
 - Consider the effects of reactant impurities and manufacturing defects on the performance of the cell over time (i.e., degradation).
 - Evaluate the electro-impedance spectra using a linearized, state-space version of the model.
 - Perform trade-off studies for system design, for example to establish the optimal air supply, humidification, or thermal management. It may be possible to use the acausal nature of the model to directly determine some control set points. In theory, the voltage or efficiency could be specified instead of a set point such as outlet pressure. The simulation would yield the pressure necessary to achieve the voltage or efficiency.
- Combine the fuel cell model with models from `Modelica.Fluid` to describe an entire fuel cell system.

- Use the fuel cell model for optimal model-based control of a fuel cell system. It may be possible to use a linearized version of the model. Tools are available to automatically linearize declarative models.
- Extend the modeling framework to complex grids (beyond rectangular cubic). This may become more feasible in Modelica as the language evolves, or it may be best done with offline tools to generate Modelica code.
- Create a 3D electrochemical/fluid package for the Modelica Standard Library. This could complement the existing 1D `Modelica.Fluid` package just as the 3D `Modelica.Mechanical.Multibody` packages complements the 1D `Modelica.Mechanical.Rotational` and `Modelica.Mechanical.Translational` packages.
- Enhancements to Modelica language and tools:
 - Add the capability to label bus (expandable) connections according to a string variable. This would simplify some of the models of the fuel cell library.
 - Leverage the repetitive structure of a model like the segmented fuel cell to reduce the translation time, reduce the simulation time by using parallel processing, and allow larger models by managing memory more efficiently during simulation.

7.4 Final Comments

A large amount of work is required to develop EOO models beyond the well-established domains. However, the potential rewards are also large in terms of not only the usefulness of EOO models, but also the knowledge that comes with creating the models. The structural and mathematical constraints of EOO language seem to force a thorough understanding of the relevant physics. This work challenges us to ask the questions posed by Willems [9]:

Did we, system theorists, get the physics right? Do our basic model structures adequately translate physical reality? Does the way in which we view interconnections respect the physics?

RELATED THEORY

This appendix contains derivations of various physical laws that draw on equations and concepts from multiple sections of Chapter 3. For a complete list of the traditional physical laws and concepts discussed in this dissertation, please see Table 3.1.

A.1 Darcy's Law

The model is consistent with Darcy's law, which describes the flow of fluid through a porous medium. To relate the model to Darcy's law, we begin with the translational diffusive exchange of a fluid species (Equation 3.55, written here with velocity on the left):

$$\phi_E - \phi_j = \frac{\mu_j}{k_{Ej}N_j} m \dot{\Phi}_{DEj} \quad (\text{A.1})$$

The subscript E refers to the exchange interface (let there be only one) and the subscript j refers to the species. If the species is in contact with a stationary solid that has zero mobility, then the mediation velocity (ϕ_E) is zero.

$$\phi_j = -\frac{\mu_j}{k_{Ej}N_j} m \dot{\Phi}_{DEj} \quad (\text{A.2})$$

The fluid may contain multiple species, but we will assume that their velocities are equal ($\phi_j = \phi$). After summing the equation over the fluid species j ,

$$\phi \sum_j \frac{k_{Ej}N_j}{\mu_j} = -m \dot{\Phi}_{DE} \quad (\text{A.3})$$

where $m \dot{\Phi}_{DE}$ is the total diffusive exchange force on the fluid species.

We will assume that the discretization is coarse enough that the solid appears uniformly distributed within the region. Then, we may neglect the shear force. If we also assume isochoric steady state, steady flow (SSSF) without body forces or chemical reactions, the conservation of translational momentum (Equation 3.28) reduces to

$$A \Delta p_j = m \dot{\Phi}_{DEj} + \sum m \dot{\Phi}_{D \perp j} \quad (\text{A.4})$$

where Δp_j is the difference in the partial pressures and $\sum m\dot{\Phi}_{D\perp j}$ is the sum of the normal forces—both of species j along the transport axis (different subscript notation than Equation 3.28). We will let p' denote the sum of the thermodynamic pressure (p) and the nonequilibrium pressure ($\pm m\dot{\Phi}_{D\perp j}/A$) over all of the fluid species.ⁱ Therefore,

$$A\Delta p' = m\dot{\Phi}_{DE} \quad (\text{A.5})$$

The previous equation can be used to replace the diffusive exchange force in Equation A.3 with a pressure difference.

$$\phi \sum_j \frac{k_{Ej}N_j}{\mu_j} = -A\Delta p' \quad (\text{A.6})$$

The volumetric flux is the product of the porosity and the velocity: $J = \varepsilon\phi$.ⁱⁱ Therefore,

$$J \sum_j \frac{k_{Ej}N_j}{\mu_j} = -\varepsilon A\Delta p' \quad (\text{A.7})$$

As a differential equation in three dimensions (assuming the same generalized resistance in each direction),

$$J \sum_j \frac{k_{Ej}N_j}{\mu_j} = -\varepsilon V \nabla p \quad (\text{A.8})$$

For now, we will establish the Darcy permeability as

$$\kappa = \frac{\varepsilon V}{\zeta \sum_j \frac{k_{Ej}N_j}{\mu_j}} \quad (\text{A.9})$$

and return to this later. Therefore,

$$J = -\kappa \zeta \nabla p \quad (\text{A.10})$$

which is Darcy's law [68, 168], where the fluidity, ζ , is the reciprocal of dynamic viscosity.

We will now consider the relation between the permeability and the parameters of the model for a basic case: a fluid that contains a single species. Equation A.9 reduces to

$$\kappa = \frac{\mu v}{\zeta k_E} \quad (\text{A.11})$$

ⁱAs discussed in Section 3.10.2, the pressure that is commonly used in the literature is generally the sum of the thermodynamic and nonequilibrium pressures in the model.

ⁱⁱThis is the reason for volumetric factor in Equation 3.123. There, the normal boundary velocity, $\phi_{\perp i}$, is the volumetric flux.

where the specific volume (v , the reciprocal of concentration) is related to the amount of the fluid species, volume of the region, and the porosity by $v = \varepsilon V/N$. In the model, the mobility (μ) and the fluidity (ζ) are different properties, but can be related under the assumptions of kinetic theory (see Sections 3.5–3.7). Due to Equations 3.56 and 3.128, their ratio is

$$\frac{\mu}{\zeta} = \frac{8\pi}{9} \frac{\lambda^2}{v} \quad (\text{A.12})$$

The mean free path (λ) is defined by Equation 3.69, and it follows that

$$\frac{\mu}{\zeta} = \frac{2v}{9\pi d^4 q^2} \quad (\text{A.13})$$

where d is the radius specific diameter of particles and q is the amount of material that represents one particle. Therefore, Equation A.11 can be written as

$$\kappa = \frac{2\pi}{9k_E} \left(\frac{v}{\pi d^2 q} \right)^2 \quad (\text{A.14})$$

The base of the square is the ratio of the specific volume of the fluid species to the specific intercept area of the fluid particles.

As an example, we will consider nitrogen as an ideal gas at 25 °C and 1 atm. We will assume that the diameter of a N_2 molecule is 420 pm—the sum of the bond length between nitrogen atoms (110 pm) and the van der Waals diameter of a nitrogen atom (310 pm).ⁱⁱⁱ We will also assume that the adjustment factor, k_E , is one. Given these assumptions, the permeability (κ) is $3.75 \times 10^{-15} \text{ m}^2$. In reality, the permeability will also depend on the solid material (porous media) and its structure; that dependence would appear in the adjustment factor. However, it is noteworthy that this rough calculation is within the expected range of 1.7×10^{-17} to $2.6 \times 10^{-12} \text{ m}^2$ [203].

A.2 Maxwell-Stefan Equations

The model encompasses Maxwell-Stefan multi-component diffusion but is more general with respect to transient behavior and the organization of the interactions. Regarding transients, the Maxwell-Stefan equations place algebraic constraints on the relationship of configuration velocities. The model does too if transient momentum storage is disabled (an option

ⁱⁱⁱThe lengths are from <http://cccbdb.nist.gov/exp2.asp?casno=7727379> and <http://en.wikipedia.org/wiki/Nitrogen>, both accessed Sep. 26, 2013.

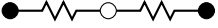
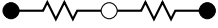

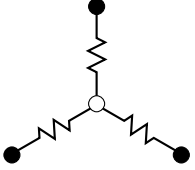
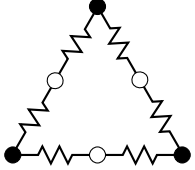
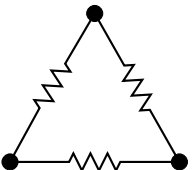
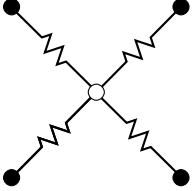
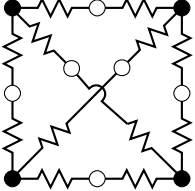
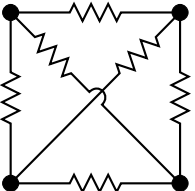
of the implementation in Chapter 4). Otherwise, the model establishes a system of differential equations that lead to these interrelationships over time. The transient option has the advantage that it avoids the nonlinear nature of the Maxwell-Stefan equations [204, 205]. Regarding the organization of the interactions, the model generalizes the Maxwell-Stefan equations so that interactions can be added among any group of species—not only pairs. In fact, the model allows connections among any group of configurations, regardless of their phase. This is useful in order to add drag between fluid species and solid species, as will be discussed at the end of this section. The Maxwell-Stefan equations are typically applied within a single phase, although there are various methods to fluid-solid interactions (e.g. the dusty gas model).

Table A.1 shows the organization of the Maxwell-Stefan equations and two possible organizations of the model for multi-component systems. Each filled circle represents the velocity of a configuration and each outlined circle represents a mediation velocity. The resistors represent friction. By default, the model considers one interaction among all species in all phases. These are the “hub” arrangements shown in the first graphic column. The model can be modified to produce the binary arrangements shown in the second graphic column by adding and removing connections in the diagram layer of the implementation in Chapter 4. At steady state, these binary arrangements are equivalent to those of Maxwell-Stefan equations shown in the last column. Strictly, the number of configurations involved in each resistive subnetwork indicates the number of particles (one from each configuration) involved in the interaction. For example, the binary and Maxwell-Stefan arrangements represent two-particle collisions [171].

If momentum storage is enabled, the configuration velocities or filled circles of the model columns (second and third graphic columns) of Table A.1 are associated with states. The filled circles of the Maxwell-Stefan column (last column) are not. States are never associated with the mediation velocities or outlined circles, since no material or mass exists at those nodes.

In a system with n_{conf} configurations, the hub arrangement has n_{conf} generalized resistors, whereas the model’s binary arrangement has $n_{\text{conf}}(n_{\text{conf}} - 1)$ and the Maxwell-Stefan equations have $n_{\text{conf}}(n_{\text{conf}} - 1)/2$. The hub arrangement has fewer resistors than the Maxwell-Stefan equations when there are four or more configurations. This is an advantage because the model has fewer parameters to specify but also a disadvantage because there are fewer degrees of

Table A.1: Structure of the model vs. the Maxwell-Stefan equations.

# of Configurations (n_{conf})	Model		
	Hub	Binary	Maxwell-Stefan
2			
3			
4			

freedom available to match experimental data. It reduces the complexity of the model, but the effect on computational performance is likely to be small or even negligible. The model's binary arrangement has twice as many resistors as the Maxwell-Stefan equations. The extra degrees of freedom determine how the generated heat is split between the configurations. In the diagrams of the first and second graphic columns of Table A.1, each resistor and the heat it generates is directly associated with a configuration.

In order to see how the model relates analytically to the Maxwell-Stefan equations, we will sum the equation for translational diffusive exchange (3.55) over all diffusive exchange interfaces for a configuration. We will represent that sum, the total diffusive exchange force on configuration i , by $\dot{m}\Phi_{\text{DE}i}$. We can use Equation 3.58 to express the mediation velocity of each node in terms of the velocities of other configurations. With some algebraic manipulation and re-indexing,

$$\dot{m}\Phi_{\text{DE}i} = \sum_{\xi \in \Xi} \frac{\sum_{j \in \xi} \frac{k_{\xi i} k_{\xi j}}{\mu_i \mu_j} N_i N_j (\phi_j - \phi_i)}{\sum_{j \in \xi} \frac{k_{\xi j}}{\mu_j} N_j} \quad (\text{A.15})$$

where ξ is a set of the configurations connected to a mediation node (through resistors) and Ξ is the set of all of those sets ξ that include configuration i . The variables $k_{\xi i}$ and $k_{\xi j}$ are the

adjustment factors for configurations i and j with respect to the note associated with set ξ . The $j = i$ term is zero in the numerator but is generally nonzero in the denominator. The momentum balance (Equation 3.190) reduces to $m\dot{\Phi}_{DEi} = A\Delta p_i$ under the following assumptions: (1) the configuration has uniform, steady-state velocity, (2) there are no reactions or phase change, and (3) there are no body forces. Therefore,

$$A\Delta p_i = \sum_{\xi \in \Xi} \frac{\sum_{j \in \xi} \frac{k_{\xi i} k_{\xi j}}{\mu_i \mu_j} N_i N_j (\phi_j - \phi_i)}{\sum_{j \in \xi} \frac{k_{\xi j}}{\mu_j} N_j} \quad (\text{A.16})$$

That is, the difference in partial pressure balances the total drag force due to other configurations. From here on, we will use a differential volume and write the equation in three dimensions:

$$V\nabla p_i = \sum_{\xi \in \Xi} \frac{\sum_{j \in \xi} \frac{k_{\xi i} k_{\xi j}}{\mu_i \mu_j} N_i N_j (\phi_j - \phi_i)}{\sum_{j \in \xi} \frac{k_{\xi j}}{\mu_j} N_j} \quad (\text{A.17})$$

If we use the binary form of the model, we can expand and simplify the summations.

$$V\nabla p_i = \sum_{j \in \xi} \frac{\phi_j - \phi_i}{\frac{\mu_i}{k_{\xi i} N_i} + \frac{\mu_j}{k_{\xi j} N_j}} \quad (\text{A.18})$$

where ξ is the set of all configurations with which configuration i interacts. This shows that the extent of the interaction between each pair of configurations depends on the amount of material present. As stated by Taylor and Krishna, “the more molecules of both types that are present in the unit volume, the higher the [rate] of collisions will be” [171]. If either configuration is removed ($N_i \rightarrow 0$ or $N_j \rightarrow 0$), there is no force. The extent of the interaction also depends on the mobilities (μ_i and μ_j). According to kinetic theory, mobility is inversely proportional to specific mass (Equation 3.56). If either particle is (hypothetically) massless, it is infinitely mobile and there can be no collision force. The previous equation can be written as the Maxwell-Stefan equation [171].

$$\frac{\nabla p_i}{p} = \sum_{j \in \xi} \frac{\chi_i \chi_j (\phi_j - \phi_i)}{D_{ij}} \quad (\text{A.19})$$

where the binary diffusion coefficient is

$$D_{ij} = pV \chi_i \chi_j \left(\frac{\mu_i}{k_{\xi i} N_i} + \frac{\mu_j}{k_{\xi j} N_j} \right) \quad (\text{A.20})$$

The subscript ξ now represents the node associated with configurations i and j . The coefficient can also be written as

$$D_{ij} = pV \frac{\frac{\mu_i}{k_{\xi i}} N_j + \frac{\mu_j}{k_{\xi j}} N_i}{N_{\text{tot}}^2} \quad (\text{A.21})$$

where N_{tot} is the total amount of material in the phase to which the Maxwell-Stefan equations are applied. As expected, it follows that the binary diffusion coefficients are symmetric ($D_{ij} = D_{ji}$). As noted previously, there is an extra degree of freedom if we wish to determine both $\mu_i/k_{\xi i}$ and $\mu_j/k_{\xi j}$ from D_{ij} or D_{ji} . By default, the adjustment factors (e.g., $k_{\xi i}$) are assumed to be one. However, in order to match the Maxwell-Stefan binary diffusion coefficients for a set of four or more species, it is necessary to relax this assumption. Otherwise, the generalized resistances (Table A.1) will be overconstrained, even though the topology of the resistor network is not.

If we instead use the hub arrangement, Equation A.17 can be written as

$$V \nabla p_i = \frac{\sum_{j \in \xi} \frac{k_{\xi i} k_{\xi j}}{\mu_i \mu_j} N_i N_j (\phi_j - \phi_i)}{\sum_{j \in \xi} \frac{k_{\xi j}}{\mu_j} N_j} \quad (\text{A.22})$$

where ξ is the set of all species in all phases. We wish for the pressure gradient to match that of the Maxwell-Stefan equation (A.19); therefore,

$$pV \sum_{j \in \xi} \frac{\chi_i \chi_j (\phi_j - \phi_i)}{D_{ij}} = \frac{\sum_{j \in \xi} \frac{k_{\xi i} k_{\xi j}}{\mu_i \mu_j} N_i N_j (\phi_j - \phi_i)}{\sum_{j \in \xi} \frac{k_{\xi j}}{\mu_j} N_j} \quad (\text{A.23})$$

In order to isolate each diffusion coefficient, we let all configurations except for j have the same velocity as i . Then, the following constraint must be satisfied:

$$D_{ij} = pV \frac{\chi_i \chi_j}{N_i N_j} \frac{\mu_i \mu_j}{k_{\xi i} k_{\xi j}} \sum_{n \in \xi} \frac{k_{\xi n}}{\mu_n} N_n \quad (\text{A.24})$$

This can also be written as

$$D_{ij} = pV \frac{1}{N_{\text{tot}}^2} \frac{\mu_i \mu_j}{k_{\xi i} k_{\xi j}} \sum_{n \in \xi} \frac{k_{\xi n}}{\mu_n} N_n \quad (\text{A.25})$$

As expected, this gives the same value for D_{ij} as the binary arrangement if there are only two species. As mentioned previously, the mobilities are overspecified when there are more than three species. That is, an arbitrary set of binary diffusion coefficients (D_{ij}) cannot be matched

by a consistent set of mobilities. Fortunately, three species are sufficient for the primary gases in a PEMFC—hydrogen (H_2) and water (H_2O) in the anode and oxygen (O_2), nitrogen (N_2), and H_2O in the cathode.

The Maxwell-Stefan equations appear in various forms in the literature. With some assumptions, the driving force (left side of Equation A.19) may be written using a gradient of chemical potential or mole fraction instead of pressure. The drag may be written in terms of a difference in the products of mole fraction and material flux rather than a difference in velocity. Taylor and Krishna discuss these alternatives [171]. Yet there is another, more significant, point of variation that arises regarding the bulk fluid motion in the implementation of the whole system of Maxwell-Stefan equations. It deserves further discussion (below) because it provides insight into the difference between the model and typical implementations of the Maxwell-Stefan equations.

The Maxwell-Stefan equations describe the drag forces among species. The sum of all these forces must be zero, and in fact, the sum of all the n_{conf} Maxwell-Stefan equations (A.19) for a system of n_{conf} species gives

$$\sum \nabla p_i = 0 \quad (\text{A.26})$$

due to the symmetry of the binary diffusion coefficients. Yet this is unrealistic. The total pressure may be nonuniform, in which case we would expect the fluid to have some bulk motion. There are two possible resolutions: (1) arbitrarily remove one of the n_{conf} Maxwell-Stefan equations and replace it with an equation that relates the total pressure gradient and the bulk motion or (2) add a term to each of the n_{conf} Maxwell-Stefan equations that accounts for part of the total pressure gradient. Both of these methods entail additional choices. This, which is compounded by the troublesome mathematical qualities of the Maxwell-Stefan equations [197, 198, 204], has led to many implementations. Cussler stated this more bluntly: “Because of an excess of theoretical zeal, many who work in this area have nurtured a glut of alternatives” [204].

Among these alternatives is the dusty-gas model, which has well-noted shortcomings [68, 157, 197, 198].^{iv} In fact, the recent implementations may miss the original point, as stated by Kerkhof and Geboers [157]:

“The vision of Maxwell, and very explicitly of Stefan, that one cannot treat a mixture as a single fluid, has also been obscured by the successful work of more recent authors on single-component fluids [...]”

Instead of attempting to cast the binary diffusion equations into a single-component framework, the model embraces the multi-component nature. With a momentum balance and associated forces for every species, it avoids (1) the inherent asymmetry in the implementation, (2) the difficulty in determining an appropriate momentum balance for the whole mixture complete with the pressure loss due to bulk flow, and (3) the nonlinear problem in solving the individual velocities. In the model, the bulk-flow pressure loss is due to two effects: the shear forces on each configuration (spatially distributed but intra-configurational) and intermolecular drag between the fluid configurations and the solid ones in each region (inter-configurational but local). In this manner, it is possible to model pressure-driven flow through a pipe, diffusion through a porous medium, or a combination of the two.

A.3 Charge Drift and Diffusion

If we assume that material is not stored along an axis through a region, then the total (advective plus diffusive) current is uniform across the region ($\dot{N}_n = -\dot{N}_p = JA$). Equations 3.104 and 3.105 imply that

$$2J = \frac{\dot{N}_n - \dot{N}_p}{A} + \phi_n \rho_n + \phi_p \rho_p \quad (\text{A.27})$$

Using the material advection and diffusion equations (3.105 and 3.106),

$$2J = \frac{k}{\eta L} \left[(\rho_n - \rho) \left(1 + e^{-\eta V \phi_{\perp i} / 2k_i A_i} \right) + (\rho - \rho_p) \left(1 + e^{\eta V \phi_{\perp i} / 2k_i A_i} \right) \right] + (\rho_n + \rho_p) \phi \quad (\text{A.28})$$

^{iv}The dusty-gas model follows the second method of resolution by adding a portion of the loss due to the interaction with the solid, as characterized by Darcy’s law, to each of the n_{conf} Maxwell-Stefan equations. However, Weber and Newman [68] have indicated that this is not rigorously correct and the Darcy’s law itself should be introduced as the final equation. Their approach follows the first method of resolution, and as noted, the choice of the Maxwell-Stefan equation to remove is arbitrary and asymmetric.

We will assume that the material Péclet number is negligible (i.e., the concentration gradient is uniform) and the area factor (k) is one. Then,

$$J = \frac{\rho_n - \rho_p}{\eta L} + \rho \phi \quad (\text{A.29})$$

We may write the velocity in terms of electric field using the translational momentum balance, assuming that the flow is steady and that there are no forces besides the electric force ($-ZE$) and the drag force ($N\phi/\mu$). Explicitly, this is, $\phi = \mu z E$.^v Thus,

$$J = \frac{\rho_n - \rho_p}{\eta L} + \rho \mu z E \quad (\text{A.30})$$

where J is the material transport rate. Taking the limit as length goes to zero ($L \rightarrow 0$) and generalizing to multiple dimensions,

$$\mathbf{J} = \rho \mu z \mathbf{E} - \frac{1}{\eta} \nabla \rho \quad (\text{A.31})$$

We can multiply this equation by the charge number (z) to write it in terms of electrical current density (zJ). For electrons ($z = -1$),

$$z\mathbf{J} = \rho \mu \mathbf{E} + \frac{1}{\eta} \nabla \rho \quad (\text{A.32})$$

and for holes ($z = 1$),

$$z\mathbf{J} = \rho \mu \mathbf{E} - \frac{1}{\eta} \nabla \rho \quad (\text{A.33})$$

These are the charge drift/diffusion equations, which are used to describe electron and hole transport in semiconductor devices [164, 206].

A.4 Ohm's Law

Ohm's law is the limiting case of charge drift/diffusion where drift current is much larger than diffusion current. If concentration is uniform, then the charge drift/diffusion equation (A.30) reduces to

$$zJ = \rho \mu E \quad (\text{A.34})$$

^vTypically, electrical mobility is expressed in terms of charge drift velocity which is the product of the bulk material velocity and the charge number. The charge number is not explicit in that definition of mobility [206].

which is Ohm's law [164, 182, 206, 207]. The factor $\rho\mu$ is the electrical conductivity [206]. If we assume that the electric field is uniform ($w = EL$) and write this in terms of electrical current ($zI = zJA$) and electrical resistance ($R = L/A\rho\mu$),

$$w = zIR \tag{A.35}$$

which is the form of Ohm's law typically used in electrical circuit theory [5, 164, 206–208], where w is the electrical potential and zI is the electrical current. This derivation may be generalized by superimposing the effects of the appropriate charge carriers (e.g., electrons and holes) [206].

Although Fick's law (Section 3.7.1) and Ohm's law have the same form (diffusivity : concentration :: conductivity : electrical potential), the modes of material transport are different. Fick's law describes material transport when it is dominated by diffusion or agitation (from high to low concentration). Ohm's law describes material transport (of charge carriers, cast as charge transport) when it is dominated by advection or translation. It happens that for electrical devices, the rate of advection is often conveniently proportional to the electric field (high to low electrical potential).

A.5 Einstein Relation

Although it was not explicit in Chapter 3 (only a brief statement in Section 3.5), the Einstein relation is built into the approximations of the diffusive exchange coefficients. If we multiply the approximations for the mobility (Equation 3.56) and the material resistance (Equation 3.107),

$$\eta\mu = \frac{8\pi}{m} \left(\frac{\tau}{\lambda} \right)^2 \tag{A.36}$$

Using the equation for the collision interval (3.70), this can be written as

$$\eta\mu T = 1 \tag{A.37}$$

or, since the diffusion coefficient is the reciprocal of the material resistivity ($D = 1/\eta$),

$$D = \mu T \tag{A.38}$$

which is the Einstein relation [164, 206].

We can derive the Einstein relation by assuming that the transported species is an ideal gas in which material advection equals material diffusion at steady state.^{vi} This implies the following at a boundary between two regions with equal material resistivities (η) and lengths (L) along an axis:

$$L\eta\rho\phi = \rho_1 - \rho_2 \quad (\text{A.39})$$

The velocity of the gas in each region may be written in terms of the drag force applied by a stationary solid in contact with the moving particles ($\phi = -\mu\dot{m}\Phi_{\text{DE}}/N$). Assuming that the velocity ϕ and mobility μ are uniform and realizing that $\rho = N/V = N/AL$:

$$-\eta\mu\dot{m}\Phi_{\text{DE}} = A(\rho_1 - \rho_2) \quad (\text{A.40})$$

If there are no other forces, then at steady state the thermodynamic force must cancel the drag force ($0 = \dot{m}\Phi_{\text{DE}} + A(p_1 - p_2)$, from conservation of translational momentum). Therefore,

$$\eta\mu(p_1 - p_2) = \rho_1 - \rho_2 \quad (\text{A.41})$$

The differences become derivatives as the regions become small ($L \rightarrow 0$):

$$\eta\mu = \frac{\partial\rho}{\partial p} \quad (\text{A.42})$$

where the right side is a property relation. Since we have assumed that the fluid is an ideal gas,

$$\eta\mu T = 1 \quad (\text{A.43})$$

which leads to the Einstein relation as above.

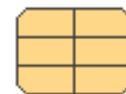
^{vi}The assumption in the typical derivation of the Einstein relation [209, 210] is material equilibrium. This imposes the opposite condition—that material advection cancels material diffusion. However, the typical derivation arrives at the same conclusion due to another negative. In particular, it uses Maxwell-Boltzmann statistics to relate the concentration of particles to the energy. In fact, Maxwell-Boltzmann statistics describe the expected number (or concentration, in proportion) of particles at an energy state rather than the energy of a particle at given concentration (i.e., external rather than internal energy).

SELECTED DOCUMENTATION AND SOURCE CODE

The following documentation has been generated from FCSys, the Modelica package discussed throughout the dissertation. It only covers selected classes; the entire library contains 244 packages, 196 models, 428 functions, and over 26,000 lines of code. An unabridged and up-to-date version of the documentation is available at <http://kdavies4.github.com/FCSys/> with a link to download the latest release version.ⁱ Here, the Modelica classes are flattened so that each appears as a section. The classes are listed in alphabetic order by their absolute paths. The subsections give general information, icons, diagrams, lists of parameters and connectors, package contents, types, constants, and source code. Where applicable, the icon (outside graphical view) of a class is shown to the right of the section heading (which is the full name of the class).

This documentation was produced by a series of programs and scripts. First, Dymola [211] was used to parse the Modelica source and produce images and HTML files. Then, `html2latex` by Thatcher and Seibert [212] and custom Python and Bash scripts were used to convert the HTML to LaTeX. The LaTeX `listings` package, maintained by Heinz and Moses [213], was used with the `dtsyntax` style file by Töpel [214] to highlight the syntax of the source code.

B.1 `FCSys.Assemblies.Cells.Examples.TestConditions`



Fuel cell test conditions

B.1.1 Information

Some conditions are taken from the outer environment model. In particular,

ⁱThe results in Chapters 5 and 6 are from version 0.2.0.

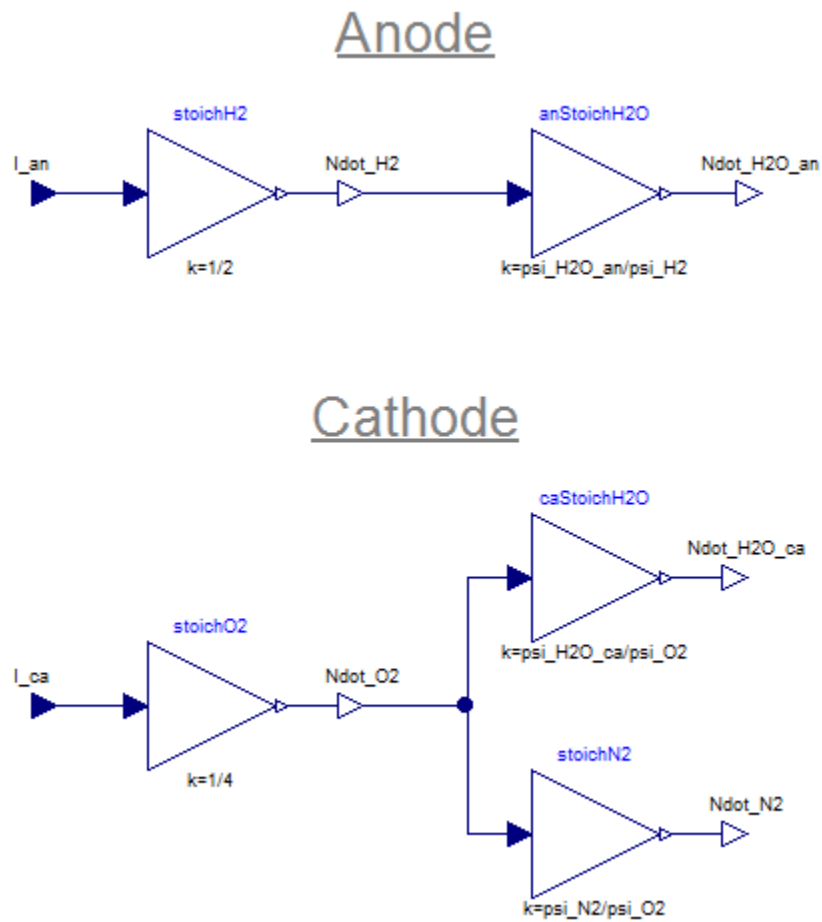


Figure B.1: Diagram of FCSys.Assemblies.Cells.Examples.TestConditions.

1. `environment.T` is used as the initial temperature throughout the cell, the temperature at each inlet, and the exterior temperature of each end plate in the yz plane.
2. `environment.p` is used as the initial pressure throughout the cell and the pressure at each outlet.
3. `environment.RH` is used as the initial relative humidity throughout the cell.
4. `environment.psi_O2_dry` is used as the dry-gas concentration of O₂ at the cathode inlet.

Figure B.1 shows the diagram of this class. Table B.1 lists its parameters. Table B.2 lists its connectors or connector variables.

This class extends from `FCSys.Assemblies.Cells.Examples.TestConditions`.

Table B.1: Parameters of `FCSys.Assemblies.Cells.Examples.TestConditions`.

Type	Name	Default	Description
—Anode—			
RealInputInternal	I_an	U.A	Equivalent current [N/T]
NumberAbsolute	anRH	0.8	Relative humidity (at inlet) [1]
—Cathode—			
RealInputInternal	I_ca	U.A	Equivalent current [N/T]
NumberAbsolute	caRH	0.5	Relative humidity (at inlet) [1]

Table B.2: Connectors of `FCSys.Assemblies.Cells.Examples.TestConditions`.

Type	Name	Description
output RealOutputInternal	Ndot_H2O_an	Rate of supply of H2O into the anode [N/T]
output RealOutputInternal	Ndot_O2	Rate of supply of O2 [N/T]
output RealOutputInternal	Ndot_H2O_ca	Rate of supply of H2O into the cathode [N/T]
output RealOutputInternal	Ndot_N2	Rate of supply of N2 [N/T]
output RealOutputInternal	Ndot_H2	Rate of supply of H2 [N/T]
Anode		

continued on the next page ...

... continued from the previous page

Type	Name	Description
input RealInputInternal	I_an	Equivalent current [N/T]
Cathode		
input RealInputInternal	I_ca	Equivalent current [N/T]

B.1.2 Modelica definition

```
model TestConditions "Fuel cell test conditions"
  extends FCSys.Icons.Record;
  // Note: This isn't a record because it contains time-varying variables.
  import FCSys.Characteristics.H2O.p_sat;

  final parameter Q.NumberAbsolute psi_sat=environment.p_sat/environment.p
    "Mole fraction of H2O at saturation";

  // Anode
  Connectors.RealInputInternal I_an(unit="N/T") = U.A "Equivalent current";
  parameter Q.NumberAbsolute anRH(
    displayUnit="%",
    max=1) = 0.8 "Relative humidity (at inlet)";
  final parameter Q.NumberAbsolute psi_H2O_an=anRH*psi_sat
    "Mole fraction of H2O at the anode inlet";
  final parameter Q.NumberAbsolute psi_H2=1 - psi_H2O_an
    "Mole fraction of H2 at the anode inlet";

  // Cathode
  Connectors.RealInputInternal I_ca(unit="N/T") = U.A "Equivalent current";
  parameter Q.NumberAbsolute caRH(
    displayUnit="%",
    max=1) = 0.5 "Relative humidity (at inlet)";
  final parameter Q.NumberAbsolute psi_H2O_ca=caRH*psi_sat
    "Mole fraction of H2O at the cathode inlet";
  final parameter Q.NumberAbsolute psi_O2=environment.psi_O2_dry*(1 -
    psi_H2O_ca) "Mole fraction of O2 at the cathode inlet";
  final parameter Q.NumberAbsolute psi_N2=(1 - environment.psi_O2_dry)*(1 -
    psi_H2O_ca) "Mole fraction of N2 at the cathode inlet";

  Modelica.Blocks.Math.Gain stoichH2(k=1/2);
  Modelica.Blocks.Math.Gain anStoichH2O(k=psi_H2O_an/psi_H2);
  Modelica.Blocks.Math.Gain stoichO2(k=1/4);
  Modelica.Blocks.Math.Gain caStoichH2O(k=psi_H2O_ca/psi_O2);
  Modelica.Blocks.Math.Gain stoichN2(k=psi_N2/psi_O2);

  Connectors.RealOutputInternal Ndot_H2O_an(unit="N/T")
    "Rate of supply of H2O into the anode";
  Connectors.RealOutputInternal Ndot_O2(unit="N/T") "Rate of supply of O2";
```



```

Connectors.RealOutputInternal Ndot_H2O_ca(unit="N/T")
  "Rate of supply of H2O into the cathode";
Connectors.RealOutputInternal Ndot_N2(unit="N/T") "Rate of supply of N2";
Connectors.RealOutputInternal Ndot_H2(unit="N/T") "Rate of supply of H2";

protected
  outer Conditions.Environment environment "Environmental conditions";

equation
  connect(stoichH2.y, Ndot_H2);
  connect(Ndot_H2, anStoichH2O.u);
  connect(anStoichH2O.y, Ndot_H2O_an);
  connect(stoichO2.y, Ndot_O2);
  connect(Ndot_O2, caStoichH2O.u);
  connect(caStoichH2O.y, Ndot_H2O_ca);
  connect(stoichN2.y, Ndot_N2);
  connect(stoichN2.u, Ndot_O2);
  connect(I_an, stoichH2.u);
  connect(I_ca, stoichO2.u);
end TestConditions;

```

B.2 FCSys.Assemblies.Cells.Examples.TestStand



Simulate the fuel cell under prescribed conditions

B.2.1 Information

Please see TestConditions regarding how the properties of the environment are used.

To run the cell with pure O₂ in the cathode (no N₂), set environment.psi_O2_dry to 100%.

Assumptions:

1. The outer surface of each end plate has uniform temperature in the yz plane.
2. No heat is conducted from the rest of the cell hardware.
3. All electronic current passes through the first segment (index [1, 1]) of each end plate in the yz plane.
4. There is no shear force on the fluid at either outlet.
5. The pressure of each gas species is uniform over each inlet.

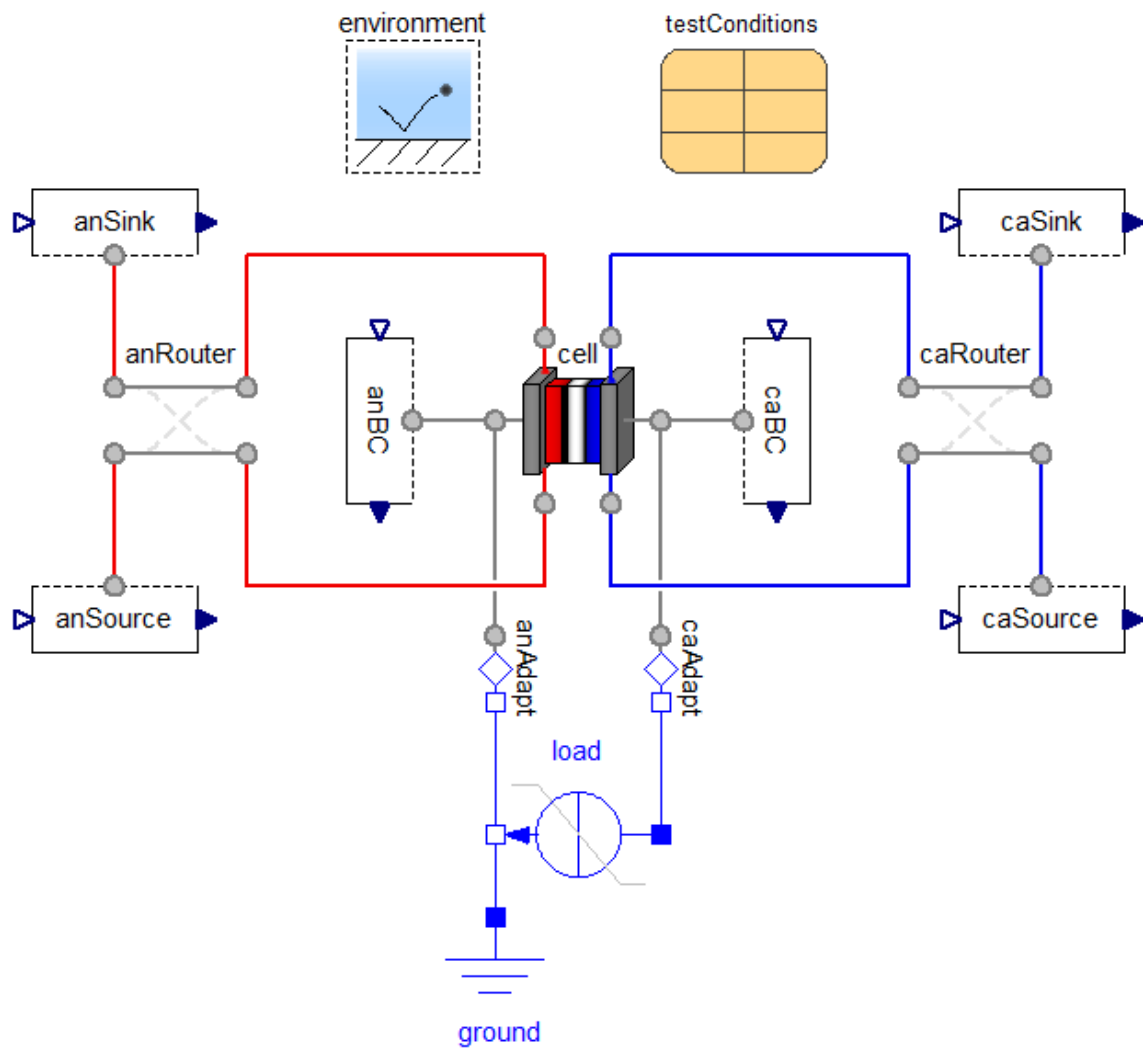


Figure B.2: Diagram of FCSys.Assemblies.Cells.Examples.TestStand.

6. The volumetric flow rates of the gas species are equal at each outlet, where the volumetric flow rate is approximated using the current at the outlet and the density of the gas within the last segment of the cell.
7. The outlet pressure is applied to the gas mixture by Dalton's law (additivity of pressure).
8. At the outlet, the liquid has the same pressure as the gas (Amagat's law).
9. There is no thermal conduction across either outlet.

Figure B.2 shows the diagram of this class.

B.3 `FCSys.Characteristics.BaseClasses.Characteristic`

Package of thermodynamic and diffusive properties

B.3.1 Information

This package is compatible with NASA CEA thermodynamic data [142] and the virial equation of state [146].

Notes regarding the constants:

- Currently, `formula` may not contain parentheses or brackets.
- d is the Van der Waals diameter or the diameter for the rigid-sphere ("billiard-ball") approximation of the kinetic theory of gases [148].
- b_c : The rows give the coefficients for the temperature intervals bounded by the values in $T_{lim\ c}$. The powers of T increase by column. By default, the powers of T for the first column are each -2, which corresponds to [142]. In that case, the dimensionalities of the coefficients are $\{L^4.M^2/(N^2.T^4), L^2.M/(N.T^2), 1, \dots\}$ for each row, where L is length, M is mass, N is particle number, and T is time. (In `FCSys`, temperature is a potential with dimension $L^2.M/(N.T^2)$; see the `Units` package.)
- B_c : As in b_c , the rows correspond to different temperature intervals. The first column is for specific enthalpy and has dimensionality $L^2.M/(N.T^2)$. The second is for specific entropy and is dimensionless. The integration constants for enthalpy are defined such that the enthalpy at 25 °C is the specific enthalpy of formation at that temperature and


reference pressure [142, p. 2]. The integration constants for specific entropy are defined such that specific entropy is absolute.

- $T_{\text{lim } c}$: The first and last entries are the minimum and maximum valid temperatures. The intermediate entries are the thresholds between rows of b_c (and B_c). Therefore, if there are n temperature intervals (and rows in b_c and B_c), then $T_{\text{lim } c}$ must have $n + 1$ entries.
- The reference pressure is p° . In the NASA CEA data [142], it is 1 bar for gases and 1 atm for condensed species. For gases, the reference state is the ideal gas at p° . For example, the enthalpy of a non-ideal (real) gas at 25 °C and p° with `ReferenceEnthalpy.zeroAt25degC` is not exactly zero.
- If the material is gaseous (`phase == Phase.gas`), then the first virial coefficient must be independent of temperature. Otherwise, the function for specific enthalpy (h) will be ill-posed. Typically, the first virial coefficient is one (or equivalently U.R), which satisfies this requirement.

Table B.3 lists the contents of this class.

This class extends from `CharacteristicEOS` (Base thermodynamic package with only the p - v - T relations).

Table B.3: Contents of the FCsys.Characteristics.BaseClasses.Characteristic package.

Name	Description
formula	Chemical formula
phase	Material phase
m	Specific mass
d	Specific diameter
z=charge(formula)	Charge number
referenceEnthalpy=ReferenceEnthalpyOfFormationAt25degC	Choice of enthalpy reference
Deltah0_f	Enthalpy of formation at 298.15 K, p° (Δh°_f)
Deltah0	$h^\circ(298.15\text{ K}) - h^\circ(0\text{ K})$ (Δh°)
h_offset=0	Additional enthalpy offset (h_{offset})
n_c=-2	Power of T for 1 st column of b_c (n_c)
T_lim_c={0,Modelica.Constants.inf}	Temperature limits for the rows of b_c and B_c ($T_{\text{lim } c}$)
b_c	Coefficients of isobaric specific heat capacity at p° as a polynomial in T (b_c)
B_c	Integration constants for specific enthalpy and entropy (B_c)
alpha=1.5*U.pi^1.5*d^2*U.q	Scaled specific intercept area
 omega	Root mean square of thermal velocity in one dimension as a function of temperature ($\omega = T/m$)






continued on the next page . . .

... continued from the previous page

Name	Description
f c_p	Isobaric specific heat capacity (c_p) as a function of temperature and pressure
f c_v	Isochoric specific heat capacity (c_v) as a function of temperature and pressure
f D	Diffusivity as a function of temperature and specific volume
f g	Gibbs potential as a function of temperature and pressure
f h	Specific enthalpy as a function of temperature and pressure
f s	Specific entropy as a function of temperature and pressure
f zeta	Continuity (ζ) as a function of temperature
f eta	Fluidity (η) as a function of temperature
f theta	Thermal resistivity (θ) as a function of temperature and specific volume
f tauprime	Phase change interval (τ') as a function of temperature and specific volume
f mu	Mobility (μ) as a function of temperature and specific volume
f nu	Thermal independity (ν) as a function of temperature and specific volume
Inherited	
p0=U.bar	Reference pressure (p°)
n_v={-1,0}	Powers of p/T and T for 1 st row and column of b_ν (n_ν)

continued on the next page ...

... continued from the previous page

Name	Description
b_v=[1]	Coefficients for specific volume as a polynomial in p/T and T (b_v)
isCompressible=...	true , if density depends on pressure
hasThermalExpansion=...	true , if density depends on temperature
n_p={n_v[1] - size(b_v, 1) + 1, n_v[2] + 1}	Powers of v and T for 1 st row and column of b_p
b_p=...	Coefficients of p as a polynomial in v and T
 dp_Tv	Derivative of pressure as defined by $p_{T,v}()$
 dv_Tp	Derivative of specific volume as defined by $v_{T,p}()$
 p_Tv	Pressure as a function of temperature and specific volume ($p_{T,v}()$)
 v_Tp	Specific volume as a function of temperature and pressure ($v_{T,p}()$)
 beta	Isothermal compressibility as a function of temperature and pressure (β)

B.4 FCSys.Characteristics.BaseClasses.Characteristic.c_p



Isobaric specific heat capacity (c_p) as a function of temperature and pressure

B.4.1 Information

For an ideal gas, this function is independent of pressure (although pressure remains as a valid input).

Tables B.4 and B.5 list the inputs and outputs of this class.

Table B.4: Inputs of FCSys.Characteristics.BaseClasses.Characteristic.c_p.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
PressureAbsolute	p	p0	Pressure [M/(L.T2)]

Table B.5: Outputs of FCSys.Characteristics.BaseClasses.Characteristic.c_p.

Type	Name	Description
CapacityThermalSpecific	c_p	Isobaric specific heat capacity [1]

B.4.2 Modelica definition

```
function c_p
  "Isobaric specific heat capacity (cp) as a function of temperature and
  pressure"
  import FCSys.Utilities.Polynomial;
  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.PressureAbsolute p=p0 "Pressure";
  output Q.CapacityThermalSpecific c_p "Isobaric specific heat capacity";
protected
  "Isobaric specific heat capacity at reference pressure as a function of
  temperature"
  import FCSys.Utilities.Polynomial;
```



```

input Q.TemperatureAbsolute T "Temperature";
output Q.CapacityThermalSpecific c0_p
  "Isobaric specific heat capacity at reference pressure";

algorithm
  c0_p := smooth(0, sum(if (T_lim_c[i] <= T or i == 1) and (T < T_lim_c[i]
    + 1]
    or i == size(T_lim_c, 1) - 1) then Polynomial.f(
      T,
      b_c[i, :],
      n_c) else 0 for i in 1:size(T_lim_c, 1) - 1));
end c0_p;

"Residual isobaric specific heat capacity for pressure adjustment"
import FCSys.Utilities.Polynomial;
input Q.TemperatureAbsolute T "Temperature";
input Q.PressureAbsolute p "Pressure";
input Integer rowLimits[2]={1,size(b_v, 1)}
  "Beginning and ending indices of rows of b_v to be included";
output Q.CapacityThermalSpecific c_p_resid
  "Temperature times the partial derivative of the integral of
    (dels/delp)_T*dp up to p w.r.t. T";

algorithm
  c_p_resid := Polynomial.F(
    p,
    {Polynomial.f(
      T,
      b_v[i, :] .* {(n_v[2] - n_v[1] + j - i)*(n_v[1] - n_v[2] + i - j +
        1)
      for j in 1:size(b_v, 2)},
      n_v[2] - n_v[1] - i) for i in rowLimits[1]:rowLimits[2]},
    n_v[1]);
  // See s_resid() in Characteristic.s for the integral of
  // (dels/delp)_T*dp.
  // This is temperature times the isobaric partial derivative of that
  // function with respect to temperature. It is zero for an ideal gas.

end c_p_resid;

algorithm
  c_p := c0_p(T) + c_p_resid(T, p) - (if phase <> Phase.gas then c_p_resid(T,
    p0) else c_p_resid(
      T,
      p0,
      {1,-n_v[1]}));
  // See the notes in the algorithm of Characteristic.s.
  // Note: [Dymond2002, p.17, eqs. 1.45 & 1.46] may be incorrect.
end c_p;

```

B.5 FCSys.Characteristics.BaseClasses.Characteristic.c_v



Isochoric specific heat capacity (c_v) as a function of temperature and pressure

B.5.1 Information

For an ideal gas, this function is independent of pressure (although pressure remains as a valid input).

Tables B.6 and B.7 list the inputs and outputs of this class.

Table B.6: Inputs of FCSys.Characteristics.BaseClasses.Characteristic.c_v.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
PressureAbsolute	p	p0	Pressure [M/(L.T2)]

Table B.7: Outputs of FCSys.Characteristics.BaseClasses.Characteristic.c_v.

Type	Name	Description
CapacityThermalSpecific	c_v	Isochoric specific heat capacity [1]

B.5.2 Modelica definition

```
function c_v
  "Isochoric specific heat capacity (cv) as a function of temperature and
  pressure"
  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.PressureAbsolute p=p0 "Pressure";
  output Q.CapacityThermalSpecific c_v "Isochoric specific heat capacity";

algorithm
  c_v := c_p(T, p) - T*dp_Tv(
    T,
    v_Tp(T, p),
    dT=1,
    dv=0)*dv_Tp(
```

```

T,
P,
dT=1,
dp=0) "[Moran2004, p. 546, eq. 11.66]";
// Note 1: This reduces to c_v = c_p - 1 for an ideal gas (where in
// FCSys 1 = U.R).
// Note 2: [Dymond2002, p.17, eqs. 1.43 & 1.44] may be incorrect.
end c_v;

```

B.6 FCSys.Characteristics.BaseClasses.Characteristic.eta



Fluidity (η) as a function of temperature

B.6.1 Information

Fluidity is defined as the reciprocal of dynamic viscosity (see <http://en.wikipedia.org/wiki/Viscosity#Fluidity>).

Although specific volume is an input to this function, the result is independent of specific volume. According to Present [148], this independence very accurately matches the measured fluidity of gases. However, the fluidity varies by species and generally falls more rapidly with temperature than indicated [148, p. 41].

This function is based on the kinetic theory of gases under the following assumptions [148]:

1. The particles are smooth and rigid but elastic spheres with identical radii. This is the “billiard-ball” assumption, and it implies that the collisions are instantaneous and conserve kinetic energy.
2. Between collisions particles have no influence on one another.
3. The mean free path, or average distance a particle travels between collisions, is much larger than the diameter of a particle.
4. The properties carried by a particle depend only on those of the last particle with which it collided.
5. The speeds of the particles follow the Maxwell-Boltzmann distribution.

Tables B.8 and B.9 list the inputs and outputs of this class.

Table B.8: Inputs of `FCSys.Characteristics.BaseClasses.Characteristic.eta`.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
VolumeSpecific	v	298.15*U.K/p0	Specific volume [L3/N]

Table B.9: Outputs of `FCSys.Characteristics.BaseClasses.Characteristic.eta`.

Type	Name	Description
Fluidity	eta	Fluidity [L.T/M]

B.6.2 Modelica definition

```

replaceable function eta
  "Fluidity as a function of temperature"

  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.VolumeSpecific v=298.15*U.K/p0 "Specific volume";
  // Note: Specific volume isn't used here but is included for generality.
  output Q.Fluidity eta "Fluidity";

algorithm
  eta := alpha/(m*omega(T));
end eta;

```

B.7 `FCSys.Characteristics.BaseClasses.Characteristic.h`



Specific enthalpy as a function of temperature and pressure

B.7.1 Information

For an ideal gas, this function is independent of pressure (although pressure remains as a valid input).

Tables B.10 and B.11 list the inputs and outputs of this class.

Table B.10: Inputs of `FCSys.Characteristics.BaseClasses.Characteristic.h`.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
PressureAbsolute	p	p0	Pressure [M/(L.T2)]

Table B.11: Outputs of `FCSys.Characteristics.BaseClasses.Characteristic.h`.

Type	Name	Description
Potential	h	Specific enthalpy [L2.M/(N.T2)]

B.7.2 Modelica definition

```

function h
  "Specific enthalpy as a function of temperature and pressure"
  import FCSys.Utilities.Polynomial;
  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.PressureAbsolute p=p0 "Pressure";
  output Q.Potential h "Specific enthalpy";

protected

  "Return h0 as a function of T using one of the temperature intervals"
  annotation(derivative=dh0_i);
  import FCSys.Utilities.Polynomial;
  input Q.TemperatureAbsolute T "Temperature";
  input Integer i "Index of the temperature interval";
  output Q.Potential h0
    "Specific enthalpy at given temperature relative to enthalpy of
     formation at 25 degC, both at reference pressure";

algorithm
  h0 := Polynomial.F(
    T,
    b_c[i, :],
    n_c) + B_c[i, 1];
  // This is the integral of c0_p*dT up to T at p0. The lower bound is the
  // enthalpy of formation (of ideal gas, if the material is gaseous) at
  // 25 degC [McBride2002, p. 2].

end h0_i;

"Derivative of h0_i"

```

```

// Note: This function is necessary for Dymola 7.4 to differentiate h().
import FCSys.Utilities.Polynomial;
input Q.TemperatureAbsolute T "Temperature";
input Integer i "Index of the temperature interval";
input Q.Temperature dT "Derivative of temperature";
output Q.Potential dh0
  "Derivative of specific enthalpy at reference pressure";

algorithm
  dh0 := Polynomial.f(
    T,
    b_c[i, :],
    n_c)*dT;

end dh0_i;

"Residual specific enthalpy for pressure adjustment for selected rows of
b_v"
import FCSys.Utilities.Polynomial;
input Q.TemperatureAbsolute T "Temperature";
input Q.PressureAbsolute p "Pressure";
input Integer rowLimits[2]={1,size(b_v, 1)}
  "Beginning and ending indices of rows of b_v to be included";
output Q.Potential h_resid
  "Integral of (delh/delp)_T*dp up to p with zero integration constant
  (for selected rows)";

algorithm
  h_resid := Polynomial.F(
    p,
    {Polynomial.f(
      T,
      b_v[i, :] .* {n_v[1] - n_v[2] + i - j + 1 for j in 1:size(b_v, 2)},
      n_v[2] - n_v[1] - i + 1) for i in rowLimits[1]:rowLimits[2]},
    n_v[1] + rowLimits[1] - 1);
  // Note: The partial derivative (delh/delp)_T is equal to v +
  // T*(dels/delp)_T by definition of enthalpy change (dh = T*ds + v*dp)
  // and then to v - T*(delv/delT)_p by applying the appropriate Maxwell
  // relation, (dels/delp)_T = -(delv/delT)_p.
  // Note: This is zero for an ideal gas.

end h_resid;

algorithm
  h := smooth(1, sum(if (T_lim_c[i] <= T or i == 1) and (T < T_lim_c[i + 1]
    or
    i == size(b_c, 1)) then h0_i(T, i) else 0 for i in 1:size(b_c, 1))) + (if
  referenceEnthalpy == ReferenceEnthalpy.zeroAtOK then Deltah0 else 0) -
  (if
  referenceEnthalpy == ReferenceEnthalpy.enthalpyOfFormationAt25degC then 0
  else Deltah0_f) + h_offset + h_resid(T, p) - (if phase <> Phase.gas then
  h_resid(T, p0) else h_resid(
  T,
  p0,

```

```

    {1,-n_v[1]}}));
// The last two terms adjust for the actual pressure relative to the
// reference. In general, the lower limit of the integral of
// (delh/delp)_T*dp is the reference pressure (p0). However, if the
// material is gaseous, then the reference is the corresponding ideal gas.
// In that case, the lower limit of the real gas terms of the integral is
// p=0, where a real gas behaves as an ideal gas. See [Rao1997, p. 271].
end h;

```

B.8 FCSys.Characteristics.BaseClasses.Characteristic.mu



Mobility (μ) as a function of temperature and specific volume

B.8.1 Information

This function is based on the kinetic theory of gases under the following assumptions [148]:

1. The particles are smooth and rigid but elastic spheres with identical radii. This is the “billiard-ball” assumption, and it implies that the collisions are instantaneous and conserve kinetic energy.
2. Between collisions particles have no influence on one another.
3. The mean free path, or average distance a particle travels between collisions, is much larger than the diameter of a particle.
4. The properties carried by a particle depend only on those of the last particle with which it collided.
5. The speeds of the particles follow the Maxwell-Boltzmann distribution.

Also, it is assumed that the Einstein relation applies.

Tables B.12 and B.13 list the inputs and outputs of this class.

Table B.12: Inputs of FCSys.Characteristics.BaseClasses.Characteristic.mu.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
VolumeSpecific	v	298.15*U.K/p0	Specific volume [L3/N]

Table B.13: Outputs of `FCSys.Characteristics.BaseClasses.Characteristic.mu`.

Type	Name	Description
Mobility	mu	Mobility [N.T/M]

B.8.2 Modelica definition

```

replaceable function mu
  "Mobility as a function of temperature and specific volume"

  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.VolumeSpecific v=298.15*U.K/p0 "Specific volume";
  output Q.Mobility mu "Mobility";

algorithm
  mu := v/(m*alpha*omega(T));
end mu;

```

B.9 `FCSys.Characteristics.BaseClasses.Characteristic.nu`



Thermal independity (ν) as a function of temperature and specific volume

B.9.1 Information

Thermal independity describes the extent to which an exchange of thermal energy between species causes or requires a temperature difference.

This function is based on the kinetic theory of gases under the following assumptions [148]:

1. The particles are smooth and rigid but elastic spheres with identical radii. This is the “billiard-ball” assumption, and it implies that the collisions are instantaneous and conserve kinetic energy.
2. Between collisions particles have no influence on one another.
3. The mean free path, or average distance a particle travels between collisions, is much larger than the diameter of a particle.

4. The properties carried by a particle depend only on those of the last particle with which it collided.
5. The speeds of the particles follow the Maxwell-Boltzmann distribution.

Also, it is assumed that the Einstein relation applies.

Tables B.14 and B.15 list the inputs and outputs of this class.

Table B.14: Inputs of `FCSys.Characteristics.BaseClasses.Characteristic.nu`.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
VolumeSpecific	v	298.15*U.K/p0	Specific volume [L3/N]

Table B.15: Outputs of `FCSys.Characteristics.BaseClasses.Characteristic.nu`.

Type	Name	Description
TimeAbsolute	nu	Thermal independity [T]

B.9.2 Modelica definition

```

replaceable function nu
  "Thermal independity as a function of temperature and specific volume"

  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.VolumeSpecific v=298.15*U.K/p0 "Specific volume";
  output Q.TimeAbsolute nu "Thermal independity";

algorithm
  nu := v/(c_p(T, p_Tv(T, v))*alpha*omega(T));
end nu;

```

B.10 `FCSys.Characteristics.BaseClasses.Characteristic.s`



Specific entropy as a function of temperature and pressure

B.10.1 Information

Tables B.16 and B.17 list the inputs and outputs of this class.

Table B.16: Inputs of `FCSys.Characteristics.BaseClasses.Characteristic.s`.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
PressureAbsolute	p	p0	Pressure [M/(L.T2)]

Table B.17: Outputs of `FCSys.Characteristics.BaseClasses.Characteristic.s`.

Type	Name	Description
NumberAbsolute	s	Specific entropy [1]

B.10.2 Modelica definition

```
function s
  "Specific entropy as a function of temperature and pressure"
  import FCSys.Utilities.Polynomial;
  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.PressureAbsolute p=p0 "Pressure";
  output Q.NumberAbsolute s "Specific entropy";

protected

  "Return s0 as a function of T using one of the temperature intervals"
  annotation(derivative=ds0_i);
  import FCSys.Utilities.Polynomial.F;
  input Q.TemperatureAbsolute T "Temperature";
  input Integer i "Index of the temperature interval";
  output Q.NumberAbsolute s0
    "Specific entropy at given temperature and reference pressure";

algorithm
  s0 := F(
    T,
    b_c[i, :],
    n_c - 1) + B_c[i, 2];
  // This is the integral of c0_p/T*dT up to T at p0 with the absolute
  // entropy at the lower bound [McBride2002, p. 2].

end s0_i;
```

```

"Derivative of s0_i"
// Note: This function is necessary for Dymola 7.4 to differentiate s().
import FCSys.Utilities.Polynomial.f;
input Q.TemperatureAbsolute T "Temperature";
input Integer i "Index of the temperature interval";
input Q.Temperature dT "Derivative of temperature";
output Q.Number ds0
  "Derivative of specific entropy at given temperature and reference
    pressure";

algorithm
  ds0 := f(
    T,
    b_c[i, :],
    n_c - 1)*dT;
end ds0_i;

"Residual specific entropy for pressure adjustment for selected rows of b_v"
annotation(derivative=ds_resid);
input Q.TemperatureAbsolute T "Temperature";
input Q.PressureAbsolute p "Pressure";
input Integer rowLimits[2]={1,size(b_v, 1)}
  "Beginning and ending indices of rows of b_v to be included";
output Q.NumberAbsolute s_resid
  "Integral of (dels/delp)_T*dp up to p with zero integration constant
    (for selected rows)";

algorithm
  s_resid := Polynomial.F(
    p,
    {Polynomial.f(
      T,
      b_v[i, :] .* {n_v[1] - n_v[2] + i - j for j in 1:size(b_v, 2)},
      n_v[2] - n_v[1] - i) for i in rowLimits[1]:rowLimits[2]},
    n_v[1] + rowLimits[1] - 1);
  // Note: According to the Maxwell relations,
  // (dels/delp)_T = -(delv/delT)_p.

end s_resid;

"Derivative of s_resid"
// Note: This function is necessary for Dymola 7.4 to differentiate s().
input Q.TemperatureAbsolute T "Temperature";
input Q.PressureAbsolute p "Pressure";
input Integer rowLimits[2]={1,size(b_v, 1)}
  "Beginning and ending indices of rows of b_v to be included";
input Q.Temperature dT "Derivative of temperature";
input Q.Pressure dp "Derivative of pressure";
output Q.Number ds_resid
  "Derivative of integral of (dels/delp)_T*dp up to p with zero
    integration constant (for selected rows)";

algorithm

```

```

ds_resid := Polynomial.dF(
  p,
  {Polynomial.df(
    T,
    b_v[i, :] .* {n_v[1] - n_v[2] + i - j for j in 1:size(b_v, 2)},
    n_v[2] - n_v[1] - i,
    dT) for i in rowLimits[1]:rowLimits[2]},
  n_v[1] + rowLimits[1] - 1,
  dp);
end ds_resid;

algorithm
s := smooth(1, sum(if (T_lim_c[i] <= T or i == 1) and (T < T_lim_c[i + 1]
or
i == size(b_c, 1)) then s0_i(T, i) else 0 for i in 1:size(b_c, 1))) +
s_resid(T, p) - (if phase <> Phase.gas then s_resid(T, p0) else s_resid(
T,
p0,
{1,-n_v[1]}));
// The first term gives the specific entropy at the given temperature and
// reference pressure (p0). The following terms adjust for the actual
// pressure (p) by integrating (dels/delp)_T*dp. In general, the
// integration is from p0 to p. However, for gases the reference state
// is the ideal gas at the reference pressure. Therefore, the lower
// integration limit for the higher-order real gas terms (i.e., terms with
// power of p greater than -1) is p=0. This is pressure limit at which a
// real gas behaves as an ideal gas, and it implies that those terms are
// zero. See [Rao1997, p. 272]. Note that the first V_m inside the curly
// brackets of the related eq. 1.47 in [Dymond2002, p. 17] should be a
// subscript rather than a multiplicative factor.
// Note: If the phase is gas, the virial equation of state (as defined by
// b_v and n_v) must include an ideal gas term (v = ... + f(T)/p +
// ...). Otherwise, an indexing error will occur.
end s;

```

B.11 FCSys.Characteristics.BaseClasses.Characteristic.tauprime



Phase change interval (τ') as a function of temperature and specific volume

B.11.1 Information

This function is based on the kinetic theory of gases under the following assumptions [148]:

1. The particles are smooth and rigid but elastic spheres with identical radii. This is the “billiard-ball” assumption, and it implies that the collisions are instantaneous and conserve kinetic energy.
2. Between collisions particles have no influence on one another.
3. The mean free path, or average distance a particle travels between collisions, is much larger than the diameter of a particle.
4. The properties carried by a particle depend only on those of the last particle with which it collided.
5. The speeds of the particles follow the Maxwell-Boltzmann distribution.

Also, it is assumed that the Einstein relation applies.

Although specific volume is an input to this function, the result is independent of specific volume.

Tables B.18 and B.19 list the inputs and outputs of this class.

Table B.18: Inputs of `FCSys.Characteristics.BaseClasses.Characteristic.tauprime`.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
VolumeSpecific	v	298.15*U.K/p0	Specific volume [L3/N]

Table B.19: Outputs of `FCSys.Characteristics.BaseClasses.Characteristic.tauprime`.

Type	Name	Description
TimeAbsolute	tauprime	Phase change interval [T]

B.11.2 Modelica definition

```
replaceable function tauprime
  "Phase change interval as a function of temperature and specific volume"
  extends Modelica.Icons.Function;
```

```

input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
input Q.VolumeSpecific v=298.15*U.K/p0 "Specific volume";
output Q.TimeAbsolute tauprime "Phase change interval";

algorithm
  tauprime := v/(alpha*omega(T));
end tauprime;

```

B.12 FCSys.Characteristics.BaseClasses.Characteristic.theta



Thermal resistivity (θ) as a function of temperature and specific volume

B.12.1 Information

This function is based on the kinetic theory of gases under the following assumptions [148]:

1. The particles are smooth and rigid but elastic spheres with identical radii. This is the “billiard-ball” assumption, and it implies that the collisions are instantaneous and conserve kinetic energy.
2. Between collisions particles have no influence on one another.
3. The mean free path, or average distance a particle travels between collisions, is much larger than the diameter of a particle.
4. The properties carried by a particle depend only on those of the last particle with which it collided.
5. The speeds of the particles follow the Maxwell-Boltzmann distribution.

Tables B.20 and B.21 list the inputs and outputs of this class.

Table B.20: Inputs of FCSys.Characteristics.BaseClasses.Characteristic.theta.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
VolumeSpecific	v	298.15*U.K/p0	Specific volume [L3/N]

Table B.21: Outputs of `FCSys.Characteristics.BaseClasses.Characteristic.theta`.

Type	Name	Description
ResistivityThermal	theta	Thermal resistivity [L.T/N]

B.12.2 Modelica definition

```

replaceable function theta
  "Thermal resistivity as a function of temperature and specific volume"

  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.VolumeSpecific v=298.15*U.K/p0 "Specific volume";
  output Q.ResistivityThermal theta "Thermal resistivity";

algorithm
  theta := alpha/(c_v(T, p_Tv(T, v))*omega(T));
end theta;

```

B.13 `FCSys.Characteristics.BaseClasses.Characteristic.zeta`



Continuity (ζ) as a function of temperature

B.13.1 Information

Continuity is a property is defined in FCSys resistivity to axial compression or material storage during transport.

Tables B.22 and B.23 list the inputs and outputs of this class.

Table B.22: Inputs of `FCSys.Characteristics.BaseClasses.Characteristic.zeta`.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
VolumeSpecific	v	298.15*U.K/p0	Specific volume [L3/N]

Table B.23: Outputs of `FCSys.Characteristics.BaseClasses.Characteristic.zeta`.

Type	Name	Description
Continuity	zeta	Continuity [L.M/(N.T)]

B.13.2 Modelica definition

```

replaceable function zeta
  "Continuity as a function of temperature"

  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.VolumeSpecific v=298.15*U.K/p0 "Specific volume";
  output Q.Continuity zeta "Continuity";

algorithm
  zeta := m*omega(T);

end zeta;

```

B.14 `FCSys.Characteristics.BaseClasses.CharacteristicEOS`

Base thermodynamic package with only the p - v - T relations

B.14.1 Information






This package may be used with the assumption of ideal gas or of constant specific volume, although it is more general than that.

Notes regarding the constants:

- b_v : The powers of p/T increase by row. The powers of T increase by column. If $n_v[1] == -1$, then the rows of b_v correspond to $1, B^*T, C^*T^2, D^*T^3, \dots$, where $1, B^*, C^*$, and D^* are the first, second, third, and fourth coefficients in the volume-explicit virial equation of state [146, pp. 1–2]. Currently, virial equations of state are supported up to the fourth coefficient (D^*). If additional terms are required, review and modify the definition of b_p .
- The defaults for b_v and n_v represent ideal gas.

Table B.24 lists the contents of this class.

Table B.24: Contents of the `FCSys.Characteristics.BaseClasses.CharacteristicEOS` package.

Name	Description
<code>p0=U.bar</code>	Reference pressure (p^0)
<code>n_v={-1,0}</code>	Powers of p/T and T for 1 st row and column of b_v (n_v)
<code>b_v=[1]</code>	Coefficients for specific volume as a polynomial in p/T and T (b_v)
<code>isCompressible=...</code>	true , if density depends on pressure
<code>hasThermalExpansion=...</code>	true , if density depends on temperature
<code>n_p={n_v[1] - size(b_v, 1) + 1, n_v[2] + 1}</code>	Powers of v and T for 1 st row and column of b_p
<code>b_p=...</code>	Coefficients of p as a polynomial in v and T
 <code>dp_Tv</code>	Derivative of pressure as defined by $p_{Tv}()$
 <code>dv_Tp</code>	Derivative of specific volume as defined by $v_{Tp}()$
 <code>p_Tv</code>	Pressure as a function of temperature and specific volume ($p_{Tv}()$)
 <code>v_Tp</code>	Specific volume as a function of temperature and pressure ($v_{Tp}()$)
 <code>beta</code>	Isothermal compressibility as a function of temperature and pressure (β)

B.15 `FCSys.Characteristics.BaseClasses.CharacteristicEOS.p_Tv`



Pressure as a function of temperature and specific volume ($p_{Tv}()$)

B.15.1 Information

If the species is incompressible then $p(T, v)$ is undefined, and the function will return a value of zero.

The derivative of this function is `dp_Tv()`.

Tables B.25 and B.26 list the inputs and outputs of this class.

Table B.25: Inputs of `FCSys.Characteristics.BaseClasses.CharacteristicEOS.p_Tv`.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
VolumeSpecificAbsolute	v	298.15*U.K/p0	Specific volume [L3/N]

Table B.26: Outputs of `FCSys.Characteristics.BaseClasses.CharacteristicEOS.p_Tv`.

Type	Name	Description
PressureAbsolute	p	Pressure [M/(L.T2)]

B.15.2 Modelica definition

```
function p_Tv
  "Pressure as a function of temperature and specific volume (pT v())"
  annotation(derivative=dp_Tv);
  import FCSys.Utilities.Polynomial;
  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.VolumeSpecificAbsolute v=298.15*U.K/p0 "Specific volume";
  output Q.PressureAbsolute p "Pressure";
algorithm
  p := if isCompressible then Polynomial.f(
    v,
    {Polynomial.f(
      T,
      b_p[i, :],
      n_p[2]) for i in 1:size(b_p, 1)},
    n_p[1]) else p0;
end p_Tv;
```

B.16 `FCSys.Characteristics.BaseClasses.CharacteristicEOS.v_Tp`



Specific volume as a function of temperature and pressure ($v_{Tp}()$)

B.16.1 Information

The derivative of this function is `dv_Tp()`.

Tables B.27 and B.28 list the inputs and outputs of this class.

Table B.27: Inputs of `FCSys.Characteristics.BaseClasses.CharacteristicEOS.v_Tp`.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
PressureAbsolute	p	p0	Pressure [M/(L.T2)]

Table B.28: Outputs of `FCSys.Characteristics.BaseClasses.CharacteristicEOS.v_Tp`.

Type	Name	Description
VolumeSpecificAbsolute	v	Specific volume [L3/N]

B.16.2 Modelica definition

```
function v_Tp
  "Specific volume as a function of temperature and pressure (vT p())"
  annotation(derivative=dv_Tp);
  import FCSys.Utilities.Polynomial;
  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.PressureAbsolute p=p0 "Pressure";
  output Q.VolumeSpecificAbsolute v "Specific volume";
algorithm
  v := Polynomial.f(
    p,
    {Polynomial.f(
      T,
      b_v[i, :],
      n_v[2] - n_v[1] - i + 1) for i in 1:size(b_v, 1)},
    n_v[1]);
end v_Tp;
```

B.17 `FCSys.Characteristics.BaseClasses.CharacteristicNASA`

Thermodynamic package with diffusive properties based on NASA CEA





B.17.1 Information

The correlations for transport properties are available in [142, 179]. For more information, please see the `Characteristic` package.

Table B.29 lists the contents of this class.

This class extends from `Characteristic` (Package of thermodynamic and diffusive properties).

Table B.29: Contents of the FCSys.Characteristics.BaseClasses.CharacteristicNASA package.

Name	Description
T_lim_eta_theta={0,Modelica.Constants.inf}	Temperature limits for the rows of b_η and b_θ ($T_{\text{lim } \eta \theta}$)
b_eta	Correlation constants for fluidity (b_η)
b_theta	Correlation constants for thermal resistivity (b_θ)
 fromNASAViscosity	Return constants for fluidity given NASA CEA constants for viscosity
 fromNASAThermalConductivity	Return constants for thermal resistivity given NASA CEA constants for thermal conductivity
 eta	Fluidity (η) as a function of temperature
 theta	Thermal resistivity (θ) as a function of temperature
Inherited	
formula	Chemical formula
phase	Material phase
m	Specific mass
d	Specific diameter
z=charge(formula)	Charge number
referenceEnthalpy=ReferenceEnthalpyOfFormationAt25degC	Choice of enthalpy reference
Deltah0_f	Enthalpy of formation at 298.15 K, p° (Δh°_f)
Deltah0	$h^\circ(298.15 \text{ K}) - h^\circ(0 \text{ K})$ (Δh°)








continued on the next page ...

... continued from the previous page

Name	Description
h_offset=0	Additional enthalpy offset (h_{offset})
n_c=-2	Power of T for 1 st column of b_c (n_c)
T_lim_c={0,Modelica.Constants.inf}	Temperature limits for the rows of b_c and B_c ($T_{\text{lim } c}$)
b_c	Coefficients of isobaric specific heat capacity at p^0 as a polynomial in T (b_c)
B_c	Integration constants for specific enthalpy and entropy (B_c)
alpha=1.5*U.pi^1.5*d^2*U.q	Scaled specific intercept area
f omega	Root mean square of thermal velocity in one dimension as a function of temperature ($\omega = T/m$)
f c_p	Isobaric specific heat capacity (c_p) as a function of temperature and pressure
f c_v	Isochoric specific heat capacity (c_v) as a function of temperature and pressure
f D	Diffusivity as a function of temperature and specific volume
f g	Gibbs potential as a function of temperature and pressure
f h	Specific enthalpy as a function of temperature and pressure
f s	Specific entropy as a function of temperature and pressure
f zeta	Continuity (ζ) as a function of temperature
f tauprime	Phase change interval (τ') as a function of temperature and specific volume

continued on the next page ...

... continued from the previous page

Name	Description
 <code>mu</code>	Mobility (μ) as a function of temperature and specific volume
 <code>nu</code>	Thermal independity (ν) as a function of temperature and specific volume
<code>p0=U.bar</code>	Reference pressure (p^0)
<code>n_v={-1,0}</code>	Powers of p/T and T for 1 st row and column of b_ν (n_ν)
<code>b_v=[1]</code>	Coefficients for specific volume as a polynomial in p/T and T (b_ν)
<code>isCompressible=...</code>	true , if density depends on pressure
<code>hasThermalExpansion=...</code>	true , if density depends on temperature
<code>n_p={n_v[1] - size(b_v, 1) + 1, n_v[2] + 1}</code>	Powers of ν and T for 1 st row and column of b_p
<code>b_p=...</code>	Coefficients of p as a polynomial in ν and T
 <code>dp_Tv</code>	Derivative of pressure as defined by $p_{T,\nu} O$
 <code>dv_Tp</code>	Derivative of specific volume as defined by $\nu_{T,p} O$
 <code>p_Tv</code>	Pressure as a function of temperature and specific volume ($p_{T,\nu} O$)
 <code>v_Tp</code>	Specific volume as a function of temperature and pressure ($\nu_{T,p} O$)
 <code>beta</code>	Isothermal compressibility as a function of temperature and pressure (β)

B.18 FCSys.Characteristics.BaseClasses.CharacteristicNASA.eta



Fluidity (η) as a function of temperature

B.18.1 Information

This function is based on based on NASA CEA [178, 179]

Although specific volume is an input to this function, the result is independent of specific volume.

Tables B.30 and B.31 list the inputs and outputs of this class.

Table B.30: Inputs of FCSys.Characteristics.BaseClasses.CharacteristicNASA.eta.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
VolumeSpecific	v	298.15*U.K/U.atm	Specific volume [L3/N]

Table B.31: Outputs of FCSys.Characteristics.BaseClasses.CharacteristicNASA.eta.

Type	Name	Description
Resistivity	eta	Fluidity [L.T/N]

B.18.2 Modelica definition

```
redeclare function eta
  "Fluidity as a function of temperature"

  extends Modelica.Icons.Function;
  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.VolumeSpecific v=298.15*U.K/U.atm "Specific volume";
  // Note: Specific volume isn't used here but is included for generality.
  output Q.Resistivity eta "Fluidity";

algorithm
```

```

eta := smooth(0, exp(sum(if (T_lim_eta_theta[i] <= T or i == 1) and (T <
  T_lim_eta_theta[i + 1] or i == size(T_lim_eta_theta, 1) - 1) then
    b_eta[i, 1]
  *log(T) + (b_eta[i, 2] + b_eta[i, 3])/T)/T + b_eta[i, 4] else 0 for i in
  1:
  size(T_lim_eta_theta, 1) - 1)));
end eta;

```

B.19 FCSys.Characteristics.BaseClasses.CharacteristicNASA.theta



Thermal resistivity (θ) as a function of temperature

B.19.1 Information

This function is based on based on NASA CEA [178, 179]

Although specific volume is an input to this function, the result is independent of specific volume.

Tables B.32 and B.33 list the inputs and outputs of this class.

Table B.32: Inputs of FCSys.Characteristics.BaseClasses.CharacteristicNASA.theta.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
VolumeSpecific	v	298.15*U.K/U.atm	Specific volume [L3/N]

Table B.33: Outputs of FCSys.Characteristics.BaseClasses.CharacteristicNASA.theta.

Type	Name	Description
ResistivityThermal	theta	Thermal resistivity [L.T/N]

B.19.2 Modelica definition

```

redeclare function theta

```

```

"Thermal resistivity as a function of temperature"

extends Modelica.Icons.Function;
input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
input Q.VolumeSpecific v=298.15*U.K/U.atm "Specific volume";
// Note: Specific volume isn't used here but is included for generality.
output Q.ResistivityThermal theta "Thermal resistivity";

algorithm
theta := smooth(0, exp(sum(if (T_lim_eta_theta[i] <= T or i == 1) and (T <
  T_lim_eta_theta[i + 1] or i == size(T_lim_eta_theta, 1) - 1) then
  b_theta[i,
  1]*log(T) + (b_theta[i, 2] + b_theta[i, 3]/T)/T + b_theta[i, 4] else 0
  for
  i in 1:size(T_lim_eta_theta, 1) - 1)));
end theta;

```

B.20 FCSys.Characteristics.'C+'.Graphite

C+ graphite

B.20.1 Information

Assumptions:

1. Constant specific volume (i.e., incompressible and without thermal expansion)

Additional notes:

- The data for this species is for C rather than C⁺, with the exception of specific mass.
- The radius is from <http://en.wikipedia.org/wiki/Carbon>. See also http://en.wikipedia.org/wiki/Van_der_Waals_radius.
- The default specific volume ($v = U.\text{cc}\cdot\text{m}/(2.210\cdot U.\text{g})$) is of pyrolytic graphite at 300 K according to [172, p. 909]. Other forms are (Ibid., also at 300 K) are:
 - Amorphous carbon: $v = U.\text{cc}\cdot\text{m}/(1.950\cdot U.\text{g})$
 - Diamond (type IIa): $v = U.\text{cc}\cdot\text{m}/(3.500\cdot U.\text{g})$

For more information, please see the Characteristic package. This class extends from BaseClasses.Characteristic (Package of thermodynamic and diffusive properties).

B.21 FCSys.Characteristics.'e-'.Graphite

e- in graphite

B.21.1 Information

Assumptions:

1. There is a 1:1 ratio of free (conductance band) electrons and carbon atoms. The density of the carbon is set by 'C+'. Graphite.

This class extends from Gas (e- gas).

B.22 FCSys.Characteristics.'H+'.Ionomer

H+ in ionomer

B.22.1 Information

The initial density corresponds to the measurement by Spry and Fayer (0.95 M) in Nafion[®] at $\lambda = 12$, where λ is the number of H₂O molecules to SO₃H endgroups. At $\lambda = 22$, the density was measured at 0.54 M [215]. This class extends from Gas (H+ gas).

B.23 FCSys.Characteristics.'SO3-'.Ionomer

C₉HF₁₇O₅S- ionomer

B.23.1 Information

Assumptions:

1. Constant specific volume (i.e., incompressible and without thermal expansion)

The specific volume ($v = U.\text{cc}\cdot\text{m}/(2.00\cdot U.\text{g})$) is based on [216, p. A1327]. Note that this is approximately 1.912 M, which does not match the default density of 'H+'. Ionomer (0.95 M), but it simplifies the model by requiring only C₁₉HF₃₇O₅S⁻ (not C₁₉HF₃₇O₅S) for charge neutrality.

Additional notes:

- Most of the data for this species is for C₁₉HF₃₇O₅S rather than C₁₉HF₃₇O₅S⁻ (with the exception of specific mass).

- A form of $C_{19}HF_{37}O_5S$ is $C_7HF_{13}O_5S.(C_2F_4)_6$, which is a typical configuration of Nafion sulfonate (after hydrolysis) [217, p. 234].
- Thermodynamic data for this material is not available from [142]. The default specific heat capacity ($b_c = [4188 * U. J * m / (U. kg * U. K)]$) is based on [218, p. B472].
- According to [219], the furthest distance between two atoms of $C_{19}HF_{37}O_5S$ is 2259.8 pm and is between fluorines. The radius of F is 147 pm (<http://en.wikipedia.org/wiki/Fluorine>).
- From [219], the molecular weight of $C_{19}HF_{37}O_5S$ is 1044.214 g/mol. However, the “11” in Nafion 11x indicates a molecular weight of 1100 g/mol. According to <http://en.wikipedia.org/wiki/Nafion>, “the molecular weight of Nafion is uncertain due to differences in processing and solution morphology.”

For more information, please see the `Characteristic` package. This class extends from `BaseClasses.Characteristic` (Package of thermodynamic and diffusive properties).

B.24 `FCSys.Characteristics.H2.Gas`

H2 gas

B.24.1 Information

Notes:

- According to [219], the (center-to-center) bond length of H-H is 100.3 pm. The radius of H is from <http://en.wikipedia.org/wiki/Hydrogen>. See also http://en.wikipedia.org/wiki/Van_der_Waals_radius.
- The virial coefficients are from [146, p. 41]. The temperature range of the coefficients is [60, 500] K, but this is not enforced in the functions.

For more information, please see the `Characteristic` package. This class extends from `BaseClasses.CharacteristicNASA` (Thermodynamic package with diffusive properties based on NASA CEA).

B.25 `FCSys.Characteristics.H2O.Gas`

H2O gas

B.25.1 Information

Notes:

- The radius of H₂O is 282 pm (<http://www.lsbu.ac.uk/water/molecule.html>). Using the radius of H from <http://en.wikipedia.org/wiki/Hydrogen> and the center-to-center distance of hydrogen atoms in H₂O from [219], 156.6 pm, the radius of H₂O would be $(120 + 156.6/2)$ pm = 198.3 pm.
- The virial coefficients are from [146, p. 4]. The temperature range of the coefficients is [350, 770] K, but this is not enforced in the functions.

For more information, please see the `Characteristic` package. This class extends from `BaseClasses.CharacteristicNASA` (Thermodynamic package with diffusive properties based on NASA CEA).

B.26 `FCSys.Characteristics.H2O.Ionomer`

H2O in ionomer

B.26.1 Information

The thermodynamic data is set such that the density in equilibrium with H₂O vapor will match the Springer et al. hydration curve [64] (see `lambda_eq()`) at $\lambda = 14$ and $\lambda = 0$. Otherwise, the properties are the same as H₂O as an ideal gas.

For more information, please see the `Characteristic` package. This class extends from `Gas` (H2O gas).

B.27 `FCSys.Characteristics.H2O.Liquid`

H2O liquid

B.27.1 Information

Assumptions:

1. Constant specific volume (i.e., incompressible and without thermal expansion)

Additional notes:

- See `Characteristics.H2O.Gas` regarding the radius.
- The default specific volume ($b_v = [\text{U.cc*m}/(0.99656*\text{U.g})]$) is at 300 K based on [220].

For more information, please see the `Characteristic` package. This class extends from `BaseClasses.Characteristic` (Package of thermodynamic and diffusive properties).

B.28 FCSys.Characteristics.MobilityFactors.BaseClasses.k

Binary mobility factor

B.28.1 Information

v_A and v_B are given as inputs even though they can be calculated from T , p_A , and p_B because it may be desirable to leave v_A and v_B constant while varying p_A and p_B .

Table B.34: Inputs of FCSys.Characteristics.MobilityFactors.BaseClasses.k.

Type	Name	Default	Description
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
PressureAbsolute	p_A	U.atm	Pressure of the 1 st species [M/(L.T2)]
PressureAbsolute	p_B	U.atm	Pressure of the 2 nd species [M/(L.T2)]
VolumeSpecific	v_A	298.15*U.K/A.p0	Specific volume of the 1 st species [L3/N]
VolumeSpecific	v_B	298.15*U.K/B.p0	Specific volume of the 2 nd species [L3/N]
replaceable function pDstar	MobilityFactors.BaseClasses...	Reduced pressure-diffusivity product	
replaceable package A	FCSys.Characteristics.BaseCl...	Characteristic data of the 2 nd species	
replaceable package B	FCSys.Characteristics.BaseCl...	Characteristic data of the 2 nd species	
TemperatureAbsolute	T_crit		Geometric mean of the critical temperatures [L2.M/(N.T2)]
PressureAbsolute	p_crit		Geometric mean of the critical pressures [M/(L.T2)]

Table B.35: Outputs of `FCSys.Characteristics.MobilityFactors.BaseClasses.k`.

Type	Name	Description
replaceable function	<code>pDstar</code>	Reduced pressure-diffusivity product
replaceable package	<code>A</code>	Characteristic data of the 2 nd species
replaceable package	<code>B</code>	Characteristic data of the 2 nd species
NumberAbsolute	<code>k_Phi</code>	Binary mobility factor [1]

Tables B.34 and B.35 list the inputs and outputs of this class.

B.28.2 Modelica definition

```

function k "Binary mobility factor"
  import harmonicMean = FCSys.Utilities.Means.harmonic;

  input Q.TemperatureAbsolute T=298.15*U.K "Temperature";
  input Q.PressureAbsolute p_A=U.atm
    "Pressure of the 1st species";
  input Q.PressureAbsolute p_B=U.atm
    "Pressure of the 2nd species";
  input Q.VolumeSpecific v_A=298.15*U.K/A.p0
    "Specific volume of the 1st species";
  input Q.VolumeSpecific v_B=298.15*U.K/B.p0
    "Specific volume of the 2nd species";
  replaceable function pDstar = MobilityFactors.BaseClasses.pDstar_nonpolar
    "Reduced pressure-diffusivity product";
  replaceable package A = FCSys.Characteristics.BaseClasses.Characteristic
    "Characteristic data of the 2nd species";
  replaceable package B = FCSys.Characteristics.BaseClasses.Characteristic
    "Characteristic data of the 2nd species";

  input Q.TemperatureAbsolute T_crit
    "Geometric mean of the critical temperatures";
  input Q.PressureAbsolute p_crit "Geometric mean of the critical pressures";
  output Q.NumberAbsolute k_Phi "Binary mobility factor";

  // TODO: Add Slattery reference, cite it in the documentation.

algorithm
  k_Phi :=
    pDstar(T/T_crit)*p_crit^(2/3)*(T_crit*U.mol/U.K)^(5/6)*(U.atm/U.q)^(1
    /3)*U.cm^2/U.s/sqrt(harmonicMean({A.m,B.m})*U.g)/(p_A*B.D(T, v_A) +
    p_B*A.D
    (T, v_B))
    "Based on [Slattery1958, eq. 5] and the exchange equations in
    FCSys.Species.Species";

```

end k;

B.29 `FCSys.Characteristics.N2.Gas`

N2 gas

B.29.1 Information

Notes:

- According to [219], the (center-to-center) bond length of N-N is 145.2 pm. The radius of N is from <http://en.wikipedia.org/wiki/Nitrogen>. See also http://en.wikipedia.org/wiki/Van_der_Waals_radius.
- The virial coefficients are from [146, p. 69]. The temperature range of the coefficients is [75, 745] K, but this is not enforced in the functions. More precise virial coefficients are available from <http://www.tpub.com/content/nasa1996/NASA-96-cr4755/NASA-96-cr47550059.htm>.

For more information, please see the `Characteristic` package. This class extends from `BaseClasses.CharacteristicNASA` (Thermodynamic package with diffusive properties based on NASA CEA).

B.30 `FCSys.Characteristics.O2.Gas`

O2 gas

B.30.1 Information

Notes:

- According to [219], the (center-to-center) bond length of O-O is 128.2 pm. The radius of O is from <http://en.wikipedia.org/wiki/Oxygen>. See also http://en.wikipedia.org/wiki/Van_der_Waals_radius.
- The virial coefficients are from [146, p. 69]. The temperature range of the coefficients is [70, 495] K, but this is not enforced in the functions.

For more information, please see the `Characteristic` package. This class extends from `BaseClasses.CharacteristicNASA` (Thermodynamic package with diffusive properties based on NASA CEA).

B.31 `FCSys.Chemistry.Capillary`



Young-Laplace model for capillary pressure

B.31.1 Information

The characteristic radius (R) is the harmonic mean of the (2) principle radii of the liquid volume.

The default surface tension ($\gamma = 0.0663 \text{ N/m}$) is for saturated water at 60°C , interpolated from [172, pp. 924]. Note that the surface tension in [139] is incorrect (likely unit conversion error).

Table B.36 lists the parameters of this class. Table B.37 lists its connectors or connector variables.

Table B.36: Parameters of `FCSys.Chemistry.Capillary`.

Type	Name	Default	Description
—Geometry—			
Length	R	$U.\mu\text{m}$	Effective radius [L]
—Material properties—			
SurfaceTension	γ	$0.0663*U.\text{N}/U.\text{m}$	Surface tension [M/T ²]
Angle	θ	$140*U.\text{degree}$	Contact angle [A]

Table B.37: Connectors of `FCSys.Chemistry.Capillary`.

Type	Name	Description
Amagat	wetting	Interface to the wetting phase

continued on the next page . . .

... continued from the previous page

Type	Name	Description
Amagat	nonwetting	Interface to the nonwetting phase

B.31.2 Modelica definition

```
model Capillary "Young-Laplace model for capillary pressure"

  extends FCSys.Icons.Names.Top2;

  // Geometry
  parameter Q.Length R=U.um "Effective radius";

  // Material properties
  parameter Q.SurfaceTension gamma=0.0663*U.N/U.m "Surface tension";
  parameter Q.Angle theta=140*U.degree "Contact angle";

  // Auxiliary variables (for analysis only)
  Q.Pressure Deltap=wetting.p - nonwetting.p if environment.analysis
    "Pressure difference due to surface tension";

  Connectors.Amagat wetting "Interface to the wetting phase";
  Connectors.Amagat nonwetting "Interface to the nonwetting phase";

protected
  outer Conditions.Environment environment "Environmental conditions";

equation
  // Pressure relation
  nonwetting.p = wetting.p + 2*gamma*cos(theta)/R "Young-Laplace equation";

  // Conservation (without storage)
  0 = wetting.V + nonwetting.V "Volume";

end Capillary;
```

B.32 FCSys.Chemistry.CapillaryVolume



Volume with capillary pressure applied to the liquid

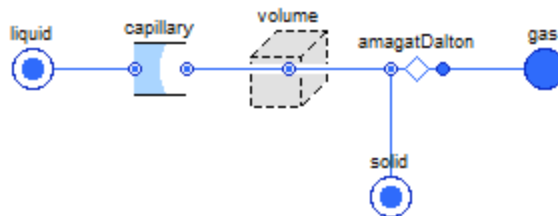


Figure B.3: Diagram of `FCSys.Chemistry.CapillaryVolume`.

B.32.1 Information

The default surface tension ($\gamma = 0.0663 \text{ N/m}$) is for saturated water at 60°C , interpolated from [172, pp. 924]. Note that the surface tension in [139] is incorrect (likely unit conversion error).

The default permeability ($\kappa = 6.46 \times 10^{-5} \text{ mm}^2$) is based on the air permeability of SGL Carbon Group Sigracet[®] 10 BA [221]. Wang et al. use $\kappa = 10^{-5} \text{ mm}^2$ [139].

The default contact angle ($\theta = 140^\circ$) is typical of the GDL measurements listed at http://www.chem.mtu.edu/cnlm/research/Movement_of_Water-in_Fuel_Cell_Electrodes.htm (accessed Nov. 22, 2103).

Figure B.3 shows the diagram of this class. Table B.38 lists its parameters. Table B.39 lists its connectors or connector variables.

Table B.38: Parameters of `FCSys.Chemistry.CapillaryVolume`.

Type	Name	Default	Description
Capillary	capillary		Capillary model —Geometry—
Volume	V		Volume [L3] —Included phases—
Boolean	inclGas	true	Gas
Boolean	inclLiquid	true	Liquid
Boolean	inclSolid	true	Solid
			—Capillary pressure—
Boolean	inclCapillary	false	Include capillary pressure

Table B.39: Connectors of FCSys.Chemistry.CapillaryVolume.

Type	Name	Description
Dalton	gas	Interface to the gas phase
Amagat	liquid	Interface to the liquid phase
Amagat	solid	Interface to the solid phase

B.32.2 Modelica definition

```

model CapillaryVolume
  "Volume with capillary pressure applied to the liquid"
  extends FCSys.Icons.Names.Top3;

  // Material properties
  parameter Q.Volume V "Volume";

  // Material properties
  parameter Boolean inclGas=true "Gas";
  parameter Boolean inclLiquid=true "Liquid";
  parameter Boolean inclSolid=true "Solid";

  // Capillary pressure
  parameter Boolean inclCapillary=false "Include capillary pressure";
  Capillary capillary if inclLiquid and inclCapillary "Capillary model";

  // Alias variables (for common terms)
  Q.Volume V_pore "Pore volume";

  // Auxiliary variables (for analysis)
  output Q.NumberAbsolute x(final stateSelect=StateSelect.never) = liquid.V/(
    gas.V + liquid.V) if inclLiquid and inclGas and environment.analysis
    "Liquid saturation";

  Connectors.Dalton gas if inclGas "Interface to the gas phase";
  Connectors.Amagat liquid if inclLiquid "Interface to the liquid phase";
  Connectors.Amagat solid "Interface to the solid phase";

  Conditions.ByConnector.Amagat.VolumeFixed volume(final V=V) if inclGas or
    inclLiquid "Fixed volume";

protected
  Conditions.Adapters.AmagatDalton amagatDalton if inclGas or (inclSolid and
    not inclLiquid)
    "Adapter between additivity of volume and additivity of gas pressure";

  outer Conditions.Environment environment "Environmental conditions";

```

```

equation
// Aliases
V = V_pore + solid.V;

if not inclCapillary then
  connect(liquid, volume.amagat)
  "Directly connect liquid to the volume (not shown in diagram)";
end if;

connect(capillary.wetting, liquid);
connect(solid, amagatDalton.amagat);
connect(volume.amagat, amagatDalton.amagat);
connect(gas, amagatDalton.dalton);
connect(capillary.nonwetting, volume.amagat);

end CapillaryVolume;

```

B.33 FCSys.Chemistry.Electrochemistry.DoubleLayer



Electrolytic double layer

B.33.1 Information

The capacitance (C) is calculated from the surface area (A), length of the gap (L), and the permittivity (ϵ) assuming that the charges are uniformly distributed over (infinite) parallel planes.

If `setVelocity` is `true`, then the material exits with the velocity of the `inert` connector. Typically, that connector should be connected to the stationary solid, in which case heat will be generated if material arrives with a nonzero velocity. That heat is rejected to the same connector.

Table B.40 lists the parameters of this class. Table B.41 lists its connectors or connector variables.

Table B.40: Parameters of FCSys.Chemistry.Electrochemistry.DoubleLayer.

Type	Name	Default	Description
Integer	n_trans		Number of components of translational momentum
Area	A	10*U.m ²	Surface area [L2]
Length	L	1e-10*U.m	Length of the gap [L]
Permittivity	epsilon	U.epsilon_0	Permittivity of the dielectric [N2.T2/(L3.M)]
replaceable package Data	Characteristics.'e'.Graphite	Material properties	
		—Assumptions—	
Boolean	setVelocity	true	Material exits at the velocity of the inert connector
Boolean	inclVolume	true	Include the amagat connector to occupy volume

Table B.41: Connectors of FCSys.Chemistry.Electrochemistry.DoubleLayer.

Type	Name	Description
replaceable package Data		Material properties
Chemical	negative	Chemical connector on the 1st side
Chemical	positive	Chemical connector on the 2nd side
Inert	inert	Translational and thermal interface with the substrate
Amagat	amagat	Connector for additivity of volume

B.33.2 Modelica definition

```

model DoubleLayer "Electrolytic double layer"
  extends FCSys.Icons.Names.Top2;

  parameter Integer n_trans(min=1,max=3)
    "Number of components of translational momentum";
  parameter Q.Area A=10*U.m^2 "Surface area";
  parameter Q.Length L=1e-10*U.m "Length of the gap";
  parameter Q.Permittivity epsilon=U.epsilon_0 "Permittivity of the
    dielectric";

  final parameter Q.Capacitance C=epsilon*A/L "Capacitance";
  replaceable package Data = Characteristics.'e-'.Graphite constrainedby
    Characteristics.BaseClasses.Characteristic "Material properties";

  parameter Boolean setVelocity=true
    "Material exits at the velocity of the inert connector";
  parameter Boolean inclVolume=true
    "Include the amagat connector to occupy volume";

  // Aliases
  Q.Potential w(
    stateSelect=StateSelect.always,
    start=0,
    fixed=true) "Electrical potential";
  Q.Current I "Material current";

  output Q.Amount Z(stateSelect=StateSelect.never) = C*w if
    environment.analysis
    "Amount of charge shifted in the positive direction";

  Connectors.Chemical negative(final n_trans=n_trans)
    "Chemical connector on the 1st side";
  Connectors.Chemical positive(final n_trans=n_trans)
    "Chemical connector on the 2nd side";

```

```

Connectors.Inert inert(final n_trans=n_trans)
    "Translational and thermal interface with the substrate";

Connectors.Amagat amagat(final V=-A*L) if inclVolume
    "Connector for additivity of volume";

protected
    outer Conditions.Environment environment "Environmental conditions";

equation
    // Aliases
    Data.z*w = positive.g - negative.g;
    I = positive.Ndot;

    // Streams
    if setVelocity then
        negative.phi = inert.phi;
        positive.phi = inert.phi;
    else
        negative.phi = inStream(positive.phi);
        positive.phi = inStream(negative.phi);
    end if;
    negative.sT = inStream(positive.sT);
    positive.sT = inStream(negative.sT);

    // Conservation
    0 = negative.Ndot + positive.Ndot "Material (no storage)";
    zeros(n_trans) = Data.m*(actualStream(negative.phi) -
        actualStream(positive.phi))
        *I + inert.mPhidot "Translational momentum (no storage)";
    der(C*w)/U.s = Data.z*I
        "Electrical energy (reversible; simplified using material conservation
        and divided by potential)";
    0 = inert.Qdot + (actualStream(negative.phi)*actualStream(negative.phi) -
        actualStream(positive.phi)*actualStream(positive.phi))*I*Data.m/2 +
        inert.phi
        *inert.mPhidot "Mechanical and thermal energy (no storage)";

end DoubleLayer;

```

B.34 FCSys.Chemistry.Electrochemistry.ElectronTransfer



Electron transfer

B.34.1 Information

fromI may help to eliminate nonlinear systems of equations if the double layer capacitance is not included.

Table B.42 lists the parameters of this class. Table B.43 lists its connectors or connector variables.

Table B.42: Parameters of FCSys.Chemistry.Electrochemistry.ElectronTransfer.

Type	Name	Default	Description
Integer	n_trans		Number of components of translational momentum
Integer	z	-1	Charge number
Current	IO	U.A	Exchange current @ 300 K [N/T]
—Chemical parameters—			
Potential	E_A	0	Activation energy [L2.M/(N.T2)]
NumberAbsolute	alpha	0.5	Charge transfer coefficient [1]
—Advanced—			
Boolean	fromI	true	Invert the Butler-Volmer equation, if $\alpha = \frac{1}{2}$

Table B.43: Connectors of FCSys.Chemistry.Electrochemistry.ElectronTransfer.

Type	Name	Description
Chemical	negative	Chemical connector on the 1st side
Chemical	positive	Chemical connector on the 2nd side
Inert	inert	Translational and thermal interface with the substrate

B.34.2 Modelica definition

```

model ElectronTransfer "Electron transfer"
  import Modelica.Math.asinh;
  extends FCSys.Icons.Names.Top2;

  parameter Integer n_trans(min=1,max=3)
    "Number of components of translational momentum";

```

```

parameter Integer z=-1 "Charge number";
parameter Q.Potential E_A=0 "Activation energy";
parameter Q.NumberAbsolute alpha(max=1) = 0.5 "Charge transfer
    coefficient";
parameter Q.Current I0=U.A "Exchange current @ 300 K";
parameter Boolean fromI=true
    "Invert the Butler-Volmer equation, if $\alpha$=";

Connectors.Chemical negative(final n_trans=n_trans)
    "Chemical connector on the 1st side";
Connectors.Chemical positive(final n_trans=n_trans)
    "Chemical connector on the 2nd side";

// Aliases
Q.TemperatureAbsolute T(start=300*U.K) "Reaction rate";
Q.Current I(start=0) "Reaction rate";
Q.Potential Deltag(start=0) "Potential difference";

Connectors.Inert inert(final n_trans=n_trans)
    "Translational and thermal interface with the substrate";

equation
// Aliases
I = positive.Ndot;
Deltag = positive.g - negative.g;
T = inert.T;

// Streams
negative.phi = inStream(positive.phi);
positive.phi = inStream(negative.phi);
negative.sT = inStream(positive.sT);
positive.sT = inStream(negative.sT);

// Reaction rate
if abs(alpha - 0.5) < Modelica.Constants.eps and fromI then
    Deltag = 2*T*asinh(0.5*exp(E_A*(1/T - 1/(300*U.K)))*I/I0);
else
    I*exp(E_A*(1/T - 1/(300*U.K))) = I0*(exp((1 - alpha)*Deltag/T) -
        exp(-alpha*
            Deltag/T));
end if;

// Conservation (without storage)
0 = negative.Ndot + positive.Ndot "Material";
zeros(n_trans) = inert.mPhidot "Translational momentum";
0 = Deltag*I + inert.Qdot "Energy";
// Note: Energy and momentum cancel among the stream terms.

end ElectronTransfer;

```

B.35 FCSys.Conditions.Adapters.ChemicalReaction



Adapter between the Chemical and Reaction connectors

B.35.1 Information

This model is used to add the stoichiometrically-weighted chemical potential of a species to the net chemical potential of a reaction. The species is produced at the stoichiometrically-weighted rate of the reaction.

For more information, please see the documentation in the Connectors package.

B.35.2 Modelica definition

```
model ChemicalReaction
  "Adapter between the Chemical and Reaction connectors"

  constant Integer n_trans(min=1,max=3)
    "Number of components of translational momentum";
  parameter Integer n "Stoichiometric coefficient";
  parameter Q.MassSpecific m "Specific mass";

  // Auxiliary variables (for analysis)
  output Q.Velocity phi[n_trans](each stateSelect=StateSelect.never) =
    actualStream(chemical.phi) if environment.analysis "Velocity of the
    stream";
  output Q.PotentialAbsolute sT(stateSelect=StateSelect.never) =
    actualStream(
      chemical.sT) if environment.analysis
    "Specific entropy-temperature product of the stream";

  Connectors.Chemical chemical(redeclare final constant Integer
    n_trans=n_trans)
    "Connector for a species in a chemical reaction";
  Connectors.Reaction reaction(final n_trans=n_trans)
    "Connector for a chemical reaction";

protected
  outer Conditions.Environment environment "Environmental conditions";

equation
  // Equal intensive properties
  reaction.g = n*chemical.g "Chemical potential";
  reaction.phi = chemical.phi "Velocity (upon outflow)";
  reaction.sT = chemical.sT
    "Specific entropy-temperature product (upon outflow)";
```

```
// Conservation (without storage)
0 = chemical.Ndot + n*reaction.Ndot "Material";
zeros(n_trans) = m*actualStream(chemical.phi)*chemical.Ndot +
    reaction.mPhidot
    "Translational momentum";
0 = actualStream(chemical.sT)*chemical.Ndot + reaction.Qdot "Energy";
end ChemicalReaction;
```

B.36 FCSys.Conditions.Environment



Environmental properties for a simulation

B.36.1 Information

Table B.44 lists the parameters of this class.

Table B.44: Parameters of FCSys.Environment.

Type	Name	Default	Description
Boolean	analysis	true	Include optional variables for analysis
			—Thermodynamics—
TemperatureAbsolute	T	298.15*U.K	Temperature [L2.M/(N.T2)]
PressureAbsolute	p	U.atm	Pressure [M/(L.T2)]
NumberAbsolute	RH	0.5	Relative humidity [1]
NumberAbsolute	psi_O2_dry	0.20946	Mole fraction of O ₂ in the dry gas [1]
			—Fields—
Acceleration	a[Axis]	{0,Modelica.Constants.g_n*U...}	Acceleration due to body forces [L/T2]
ForceSpecific	E[Axis]	{0,0,0}	Electric field [L.M/(N.T2)]

B.36.2 Modelica definition

```
record Environment "Environmental properties for a simulation"
  extends FCSys.Icons.Names.Top3;

  // Store the values of the base constants and units.
  final constant U.Bases.Base baseUnits=U.base "Base constants and units";

  parameter Boolean analysis=true "Include optional variables for analysis";

  // Thermodynamics
  parameter Q.TemperatureAbsolute T(nominal=300*U.K) = 298.15*U.K
    "Temperature";
  parameter Q.PressureAbsolute p(nominal=U.atm) = U.atm "Pressure";
  parameter Q.NumberAbsolute RH(
    displayUnit="%",
    max=1) = 0.5 "Relative humidity";
  parameter Q.NumberAbsolute psi_O2_dry(
    final max=1,
    displayUnit="%") = 0.20946
    "Mole fraction of O2 in the dry gas";
  // Value from http://en.wikipedia.org/wiki/Oxygen, accessed 2013/10/30
  final parameter Q.PressureAbsolute p_sat=Characteristics.H2O.p_sat(T)
    "Saturation pressure of H2O vapor";
  final parameter Q.PressureAbsolute p_H2O=RH*p_sat "Pressure of H2O vapor";
  final parameter Q.PressureAbsolute p_dry=p - p_H2O "Pressure of dry gases";
  final parameter Q.PressureAbsolute p_O2=psi_O2_dry*p_dry "Pressure of O2";
  final parameter Q.NumberAbsolute psi_H2O=p_H2O/p "Mole fraction of H2O";
  final parameter Q.NumberAbsolute psi_dry=1 - psi_H2O
    "Mole fraction of dry gases";

  // Fields
  parameter Q.Acceleration a[Axis]={0,Modelica.Constants.g_n*U.m/U.s^2,0}
    "Acceleration due to body forces";
  // The gravity component is positive because it's added to the transient
  // term in the Species model.
  parameter Q.ForceSpecific E[Axis]={0,0,0} "Electric field";
end Environment;
```

B.37 FCSys.Connectors

Declarative and imperative interfaces

B.37.1 Information

FCSys uses four types of declarative connectors. The chemical connectors (Chemical and Reaction) represent the diffusion of material and the advection of other quantities among configurations (i.e., species in particular phases) that react chemically within a subregion. The

inert connectors (Intra, Inter, Inert, and InertNode) describe the diffusive exchange of momentum and energy among configurations within a subregion. The mixing connectors (Amagat and Dalton) describe how species are combined within a phase and how phases are combined within a subregion. The boundary connectors (Boundary and BoundaryBus) describe the transport between neighboring regions or subregions.

Figure B.4 shows the hierarchy of the declarative connectors. The top row contains a bus connector (BoundaryBus), which expands to group the Boundary connectors of multiple species. The connectors in the middle row are flat; they build on the connectors of the bottom row by extension. Each icon on the bottom row represents one effort/flow pair, which may or may not be implemented as a separate connector. The Chemical connector also has stream variables to represent the advection of translational momentum and thermal energy.

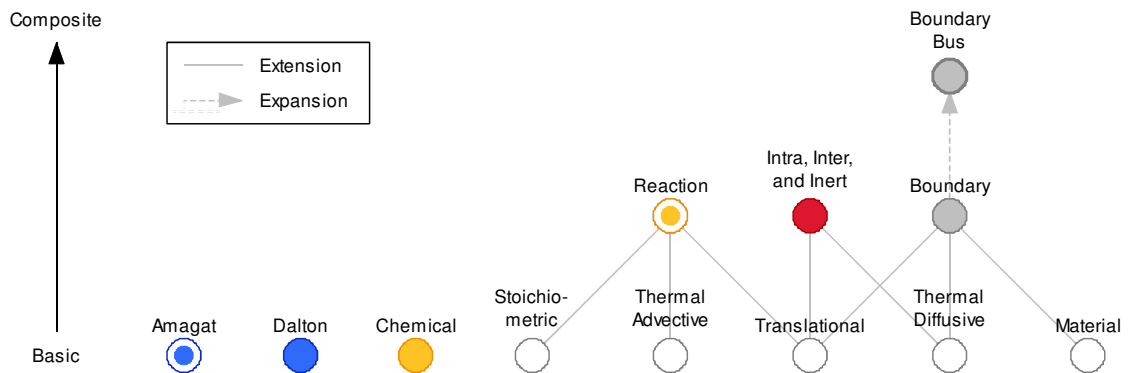


Figure B.4: Hierarchy of the connectors.

The `Chemical` connector is used for a single species in a chemical reaction or phase change process. It expresses the rate of consumption or generation of a species at a chemical potential. The `Reaction` connector is used for the chemical reaction as a whole. It has chemical potential as a flow and current as an effort (opposite designations of the `Chemical` connector). It sums the stoichiometrically weighted chemical potentials of the species participating in a reaction. Its effort variable is the rate of the reaction. `Chemical` and `Reaction` cannot be directly connected because they have opposite efforts and flows. An adapter must be used (e.g., `ChemicalReaction`), which applies the stoichiometry.

The Amagat connector imposes Amagat's law of partial volumes and is used to combine material phases within a subregion. The Dalton connector applies Dalton's law of partial pressures to mix species within a phase (e.g., N_2 and O_2 within a gas). The two cannot be directly connected because they have opposite efforts and flows. An adapter must be used (e.g., AmagatDalton).

In addition to the declarative connectors, there are connectors with inputs and outputs. The RealInput, RealInputInternal, RealInputBus, and RealInputBusInternal connectors contain only Real input variables. The RealOutput, RealOutputInternal, RealOutputBus, and RealOutputBusInternal connectors contain only Real output variables.

B.37.2 Relation to Thermodynamics

In order to describe the dynamic behavior of a physical system, a model must include conservation laws or rate balances. These equations involve the storage and flow of extensive quantities within (among configurations) and into the system. In chemical/thermal systems, the extensive quantities of interest are particle number (or mass) and energy. For the sake of simplicity, momentum will be excluded from the present discussion; assume that the fluid is macroscopically stagnant. Also assume that there is only one inlet or outlet to the system. In terms of mathematics, we have introduced four variables (2 flows and 2 quantities) but only two equations (material and energy conservation).

Two additional equations involve flow rates; these are transport equations with spatial nature—separate from the temporal conservation equations. Empirical evidence indicates that the flows are related to differences in efforts or generalized “driving forces.” The efforts are usually conjugate to the quantities with respect to energy. For the chemical/thermal system, the efforts are then chemical potential and temperature. Yet these are intensive properties—distinct from the quantities, which are extensive. So far, there are two rate balances to relate extensive quantities to flows and two transport equations (4 equations in all) and six variables (2 quantities, 2 flows, and 2 efforts or intensive properties).

One extensive quantity can be divided by the other to yield an intensive property. For example, internal energy can be divided by particle number to give internal potential (the

relationship is not as direct for chemical potential, but the concept is the same). The other equation involves the spatial extent of the system, for example, the extensive volume of the system divided by particle number to give specific volume. This introduces another variable (extensive volume); now there are six equations and seven variables.

In a Eulerian frame of reference, we assume that the extensive volume of the system is fixed (i.e., that the system is a “control volume”).ⁱⁱ If there is only one species in the system, then we can assume that it fills the entire volume (e.g., no macroscopically observable regions of vacuum). If another species is included in the system, the number of variables is doubled. All of the equations may be repeated except that the specific volume of each species is its own extensive volume or “partial volume” divided by its particle number to give “partial specific volume.” It is reasonable to assume that the sum of the partial volumes is equal to the total volume of the system (again, no voids). This is a generalization of the previous equation that set the volume of the single species equal to the volume of the system or control volume. However, now there are three volumes (of each configuration and of the system) instead of two (of the one configuration and of the system) but no additional equations.

















In general, an additional equation may be added to exchange volume between the two species such that they reach equilibrium. This could be modeled by another transport-like equation. However, in the FCSys package, it is assumed that this equilibrium already, always exists. Since we wish to impose that the sum of the two partial volumes is equal to the total volume, it is appropriate to set the flow variable to be the quantity itself (volume) rather than the rate of the quantity. Then, there is no need for another rate balance to relate the quantity to the flow; the quantity is the flow. In this case, the most appropriate effort variable is pressure. The relationship among pressure, specific volume, and temperature is given by an equation of state. This additivity-of-volume interaction occurs in the `VolumeFixed` model.

If the species are mixed, it may be more appropriate to assume that the pressures of the components of a mixture sum to the total pressure of the mixture. This additivity of pressure is described by connections of the `Dalton` connector.

ⁱⁱIn a Lagrangian frame of reference, the amount of material is fixed and thermal energy is reduced to random motion since particles are tracked directly. There are only the momentum conservation equations.

Table B.45 lists the contents of this class.

Table B.45: Contents of the FCSys.Connectors package.

Name	Description
 Chemical	Connector for a species in a reaction or phase change process
 Reaction	Connector for the combination of species in a chemical reaction
 BoundaryBus	Bus of Boundary connectors (for multiple configurations)
 Boundary	Connector to transport material, translational momentum, and thermal energy
 Amagat	Connector for additivity of volume
 Dalton	Connector for additivity of pressure
 DaltonNode	Internal node for additivity of pressure
 Intra	Connector for translational and thermal exchange among species within a phase
 Inter	Connector for translational and thermal exchange among species, regardless of phase
 Inert	Connector for translational and thermal exchange
 InertNode	Internal node for Intra and Inter
 Translational	Connector for the advection or diffusion of translational momentum
 ThermalDiffusive	Connector for the diffusion of thermal energy
 RealInput	“input Real” as a connector
 RealInputInternal	Internal “input Real” as a connector
 RealInputBus	Bus of RealInput connectors
 RealInputBusInternal	Internal bus of RealInput connectors
 RealOutput	“output Real” as a connector
 RealOutputInternal	Internal “output Real” as a connector
 RealOutputBus	Bus of RealOutput connectors
 RealOutputBusInternal	Internal bus of RealOutput connectors

B.38 FCSys.Connectors.Amagat



Connector for additivity of volume

B.38.1 Information

The concept of “additivity of volume” is defined by Amagat’s law of partial volumes, which states that the partial extensive volumes of the components of a mixture sum to the total extensive volume of the mixture [151, p. 194]. The components are assumed to each exist at the total pressure of the mixture.

This concept loses its physical meaning once the species are mixed [152]. If the species are truly mixed, then it is impossible to distinguish their particles and thus determine their partial volumes. Therefore, additivity of volume is only used for distinct phases within the same subregion—not for species within a phase. For example, if a system contains a solid phase and a gas phase, then it is assumed that the volumes of the phases are additive. Within each phase, the pressures of the species are added according to Dalton’s law (see the Dalton connector).

In order to implement Amagat’s law, this connector includes volume (not rate of volume) as a flow variable. The effort variable is pressure. This implies that the effort and flow variables are conjugates of energy (not power).

See also the Dalton connector and the documentation in the Connectors package.

Table B.46: Contents of FCSys.Connectors.Amagat.

Type	Name	Description
PressureAbsolute	p	Pressure [M/(L.T ²)]
flow Volume	V	Volume [L ³]

Table B.46 lists the contents of this class.

B.39 FCSys.Connectors.Boundary



Connector to transport material, translational momentum, and thermal energy

B.39.1 Information

This connector applies to a single species in a single phase. For multiple species or phases, use the BoundaryBus connector.

For more information, please see the documentation of the Connectors package.

Table B.47 lists the contents of this class.

This class extends from ThermalDiffusive (Connector for the diffusion of thermal energy).

Table B.47: Contents of FCSys.Connectors.Boundary.

Type	Name	Description
PressureAbsolute	p	Pressure [M/(L.T ²)]
<code>flow</code> Current	Ndot	Current [N/T]
Velocity	phi[Orient]	Velocity [L/T]
<code>flow</code> Force	mPhidot[Orient]	Force [L.M/T ²]
TemperatureAbsolute	T	Temperature [L ² .M/(N.T ²)]
<code>flow</code> Power	Qdot	Rate of thermal conduction [L ² .M/T ³]

B.40 FCSys.Connectors.Chemical



Connector for a species in a reaction or phase change process

B.40.1 Information

Note that the product of the translational `stream` variable (velocity, ϕ) and the `flow` variable (current, N) is only conserved among species with the same specific mass. This connector is not directly connected among different species. It is first converted to the `Reaction` connector using the `ChemicalReaction` adapter, which reconciles the different specific masses.

For more information, please see the documentation of the `Connectors` package.

Table B.48: Parameters of `FCSys.Connectors.Chemical`.

Type	Name	Default	Description
Integer	<code>n_trans</code>		Number of components of translational momentum

Table B.49: Contents of `FCSys.Connectors.Chemical`.

Type	Name	Description
Integer	<code>n_trans</code>	Number of components of translational momentum
Potential	<code>g</code>	Chemical potential [L2.M/(N.T2)]
<code>flow</code> Current	<code>Ndot</code>	Current [N/T]
<code>stream</code> Velocity	<code>phi[n_trans]</code>	Velocity upon outflow [L/T]
<code>stream</code> PotentialAbsolute	<code>sT</code>	Specific entropy-temperature product upon outflow [L2.M/(N.T2)]

Table B.49 lists the contents of this class. Table B.48 lists its parameters.

B.41 `FCSys.Connectors.Dalton`



Connector for additivity of pressure

B.41.1 Information

The concept of “additivity of pressure” is defined by Dalton’s law of partial pressures, which states that the partial pressures of the components of a mixture sum to the total pressure of the mixture [151, p. 192]. The components are assumed to exist at the total volume of the mixture.

In order to implement Dalton’s law, this connector includes pressure as a flow variable. The effort variable is volume. This implies that the effort and flow variables are conjugates of energy (not power).

See also the Amagat connector and the documentation of the `Connectors` package.

Table B.50: Contents of `FCSys.Connectors.Dalton`.

Type	Name	Description
Volume	V	Volume [L3]
<code>flow</code> Pressure	p	Pressure [M/(L.T2)]

Table B.50 lists the contents of this class.

B.42 `FCSys.Connectors.Inert`



Connector for translational and thermal exchange

B.42.1 Information

Please see the documentation of the `Connectors` package.

Table B.52 lists the contents of this class. Table B.51 lists its parameters.

This class extends from `Translational` (Connector for the advection or diffusion of translational momentum).

Table B.51: Parameters of `FCSys.Connectors.Inert`.

Type	Name	Default	Description
Integer	<code>n_trans</code>		Number of components of translational momentum —Initialization—
Velocity	<code>phi.start[n_trans]</code>	0	Velocity [L/T]

Table B.52: Contents of `FCSys.Connectors.Inert`.

Type	Name	Description
Integer	<code>n_trans</code>	Number of components of translational momentum
Velocity	<code>phi[n_trans]</code>	Velocity [L/T]
<code>flow</code> Force	<code>mPhidot[n_trans]</code>	Force [L.M/T ²]
TemperatureAbsolute	<code>T</code>	Temperature [L ² .M/(N.T ²)]
<code>flow</code> Power	<code>Qdot</code>	Rate of thermal conduction [L ² .M/T ³]

B.43 `FCSys.Connectors.Reaction`



Connector for the combination of species in a chemical reaction

B.43.1 Information

Please see the documentation of the `Connectors` package.

Table B.54 lists the contents of this class. Table B.53 lists its parameters.

This class extends from `Translational` (Connector for the advection or diffusion of translational momentum).

Table B.53: Parameters of `FCSys.Connectors.Reaction`.

Type	Name	Default	Description
Integer	<code>n_trans</code>		Number of components of translational momentum —Initialization—
Velocity	<code>phi.start[n_trans]</code>	0	Velocity [L/T]

Table B.54: Contents of `FCSys.Connectors.Reaction`.

Type	Name	Description
Current	<code>Ndot</code>	Rate of reaction [N/T]
<code>flow</code> Potential	<code>g</code>	Chemical potential [L ² .M/(N.T ²)]
Integer	<code>n_trans</code>	Number of components of translational momentum
Velocity	<code>phi[n_trans]</code>	Velocity [L/T]
<code>flow</code> Force	<code>mPhidot[n_trans]</code>	Force [L.M/T ²]
PotentialAbsolute	<code>sT</code>	Product of specific entropy and temperature [L ² .M/(N.T ²)]
<code>flow</code> Power	<code>Qdot</code>	Rate of thermal advection [L ² .M/T ³]

B.44 `FCSys.Connectors.ThermalDiffusive`



Connector for the diffusion of thermal energy

B.44.1 Information

Please see the documentation of the `Connectors` package.

Table B.55: Contents of `FCSys.Connectors.ThermalDiffusive`.

Type	Name	Description
TemperatureAbsolute	T	Temperature [L2.M/(N.T2)]
<code>flow</code> Power	Qdot	Rate of thermal conduction [L2.M/T3]

Table B.55 lists the contents of this class.

B.45 `FCSys.Connectors.Translational`



Connector for the advection or diffusion of translational momentum

B.45.1 Information

Please see the documentation of the `Connectors` package.

Table B.56: Parameters of `FCSys.Connectors.Translational`.

Type	Name	Default	Description
Integer	n_trans		Number of components of translational momentum

Table B.57: Contents of `FCSys.Connectors.Translational`.

Type	Name	Description
Integer	n_trans	Number of components of translational momentum
Velocity	phi[n_trans]	Velocity [L/T]
<code>flow</code> Force	mPhidot[n_trans]	Force [L.M/T2]

Table B.57 lists the contents of this class. Table B.56 lists its parameters.

B.46 `FCSys.Phases.PartialPhase`

Base model for a phase

B.46.1 Information

The scaling factor for diffusive transport (k) is a vector which directly affects the resistance associated with the transport of material, transverse translational momentum, and energy of all of the species within the phase. It can be used to introduce minor head loss or the effects of porosity or tortuosity. These effects may be anisotropic. Using Bruggeman correction [140, p. 4696], the factor (k) within a phase should be set to $\epsilon^{-1/2}$ along each axis, where ϵ is the volumetric filling ratio, or the ratio of the volume of the phase to the total volume of the subregion. The Bruggeman factor itself increases resistance by a $\epsilon^{-3/2}$, but a factor of ϵ^{-1} is included inherently.

Table B.58: Parameters of FCSys . PartialPhase.

Type	Name	Default	Description
Integer	n_spec		Number of species
Integer	n_trans	1	Number of transport axes
NumberAbsolute	k[Axis]	{1,1,1}	Length factors for transport [1] —Geometry—
NumberAbsolute	k_inter_Phi[n_inter, n_trans]	ones(n_inter, n_trans)	—Independence factors— For translational exchange with other phases [1]
NumberAbsolute	k_inter_Q[n_inter]	ones(n_inter)	For thermal exchange with other phases [1]

Table B.58 lists the parameters of this class.

B.47 FCSys.Quantities

Variables to represent physical properties

B.47.1 Information

In FCSys, the `unit` attribute of each `Real` variable actually denotes the dimension.ⁱⁱⁱ The fundamental dimensions are angle (A), length (L), mass (M), particle number (N), and time (T). These are combined according to the rules established for unit strings [193, p. 210]. Temperature and charge are derived dimensions (see the `Units` package).

The `quantity` attribute is not used since the type is the quantity. The `displayUnit` attribute is only used for quantities that imply a certain display unit.

Methods for unit checking have been established [222–224] and can, in theory, be applied to dimension checking instead.

The `Quantities` package is abbreviated as `Q` throughout the rest of FCSys. The quantities are generally named with adjectives following the noun so that the quantities are grouped when alphabetized. Some quantities are aliases to other quantities but with special implied display units. For example, `Temperature` is an alias for `Potential` with a default display unit of `K`.^{iv} Also, some quantities have minimum values (e.g., zero for `PressureAbsolute`). For more information, please see the documentation of the `Units` package.

B.47.2 Types and constants

```
type Acceleration = TypeReal (final unit="L/T2");
```

```
type Amount = TypeReal (final unit="N", min=0);
```

```
type AmountReciprocal = TypeReal (final unit="1/N", min=0)  
  "Reciprocal of amount";
```

```
type Angle = TypeReal (final unit="A");
```

```
type Angle2 = TypeReal (final unit="A2") "Solid angle";
```

ⁱⁱⁱThis misnomer is necessary because `Real` variables do not have a `dimension` attribute.

^{iv}Temperature is a potential in the chosen system of units; see Section 4.3.

```

type Area = TypeReal (final unit="L2", min=0);
type AreaSpecific = TypeReal (final unit="L2/N", min=0) "Specific area";
type Capacitance = TypeReal (final unit="N2.T2/(L2.M)", min=0);
type Density = TypeReal (final unit="N/L3", min=0);
type DensityRate = TypeReal (final unit="N/(L3.T)") "Rate of density";
type ConductanceElectrical = TypeReal (final unit="N2.T/(L2.M)", min=0)
  "Electrical conductance";
type ConductivityElectrical = TypeReal (final unit="N2.T/(L3.M)", min=0)
  "Electrical conductivity";
type Continuity = TypeReal (final unit="L.M/(N.T)", min=0);
type Current = TypeReal (final unit="N/T");
type CurrentAreic = TypeReal (final unit="N/(L2.T)") "Areic current";
type CurrentAreicAbsolute = TypeReal (final unit="N/(L2.T)", min=0)
  "Absolute areic current";
type CurrentRate = TypeReal (final unit="N/T2") "Rate of current";
type Diffusivity = TypeReal (final unit="L2/T", min=0);
type Energy = TypeReal (final unit="L2.M/T2");
type Fluidity = TypeReal (final unit="L.T/M", min=0);
type Force = TypeReal (final unit="L.M/T2");
type ForceSpecific = TypeReal (final unit="L.M/(N.T2)") "Specific force";
type Frequency = TypeReal (final unit="A/T");
type Inductance = TypeReal (final unit="L2.M/N2", min=0);
type Length = TypeReal (final unit="L", min=0);
type LengthReciprocal = TypeReal (final unit="1/L", min=0)
  "Reciprocal of length";
type LengthSpecific = TypeReal (final unit="L/N", min=0) "Specific length";
type MagneticFlux = TypeReal (final unit="L2.M/(A.N.T)") "Magnetic flux";
type MagneticFluxAreic = TypeReal (final unit="M/(A.N.T)")

```

<code>"Areic magnetic flux";</code>
<code>type MagneticFluxReciprocal = TypeReal (final unit="A.N.T/(L2.M)") "Reciprocal of magnetic flux";</code>
<code>type Mass = TypeReal (final unit="M", min=0);</code>
<code>type MassSpecific = TypeReal (final unit="M/N", min=0) "Specific mass";</code>
<code>type MassVolumic = TypeReal (final unit="M/L3", min=0) "Volumic mass";</code>
<code>type Mobility = TypeReal (final unit="N.T/M", min=0);</code>
<code>type MomentumRotational = TypeReal (final unit="L2.M/(A.T)") "Rotational momentum";</code>
<code>type Number = TypeReal (final unit="1");</code>
<code>type NumberAbsolute = TypeReal (final unit="1", min=0) "Absolute number";</code>
<code>type Permeability = TypeReal (final unit="L.M/N2", min=0);</code>
<code>type Permittivity = TypeReal (final unit="N2.T2/(L3.M)", min=0);</code>
<code>type PermittivityReciprocal = TypeReal (final unit="L3.M/(N2.T2)", min=0) "Reciprocal of permittivity";</code>
<code>type Potential = TypeReal (final unit="L2.M/(N.T2)");</code>
<code>type PotentialAbsolute = TypeReal (final unit="L2.M/(N.T2)", min=0) "Absolute potential";</code>
<code>type PotentialPerWavenumber = TypeReal (final unit="L3.M/(A.N.T2)") "Potential per wavenumber";</code>
<code>type PotentialRate = TypeReal (final unit="L2.M/(N.T3)") "Rate of potential";</code>
<code>type Power = TypeReal (final unit="L2.M/T3");</code>
<code>type PowerArea = TypeReal (final unit="L4.M/T3") "Power times area";</code>
<code>type PowerAreic = TypeReal (final unit="M/T3") "Areic power";</code>
<code>type PowerAreicPerPotential4 = TypeReal (final unit="M.T5/L8") "Areic power per 4th power of potential";</code>
<code>type PowerRadiant = TypeReal (final unit="L2.M/(A2.T3)") "Radiant power";</code>
<code>type Pressure = TypeReal (final unit="M/(L.T2)");</code>
<code>type PressureAbsolute = TypeReal (final unit="M/(L.T2)", min=0)</code>

<code>"Absolute pressure";</code>
<code>type PressureRate = TypeReal (final unit="M/(L.T3)") "Rate of pressure";</code>
<code>type PressureReciprocal = TypeReal (final unit="L.T2/M", min=0) "Reciprocal of pressure";</code>
<code>type ResistanceElectrical = TypeReal (final unit="L2.M/(N2.T)", min=0) "Electrical resistance";</code>
<code>type ResistanceThermal = TypeReal (final unit="T/N", min=0) "Thermal resistance";</code>
<code>type SurfaceTension = TypeReal (final unit="M/T2") "Surface tension";</code>
<code>type Resistivity = TypeReal (final unit="L.T/N", min=0);</code>
<code>type Time = TypeReal (final unit="T");</code>
<code>type TimeAbsolute = TypeReal (final unit="T", min=0) "Absolute time";</code>
<code>type TimeLineic = TypeReal (final unit="T/L") "Lineic time";</code>
<code>type Velocity = TypeReal (final unit="L/T");</code>
<code>type Velocity2 = TypeReal (final unit="L2/T2") "Squared velocity";</code>
<code>type VelocityAmount = TypeReal (final unit="L.N/T");</code>
<code>type Volume = TypeReal (final unit="L3", min=0);</code>
<code>type VolumeRate = TypeReal (final unit="L3/T") "Rate of volume";</code>
<code>type VolumeSpecific = TypeReal (final unit="L3/N") "Specific volume";</code>
<code>type VolumeSpecificAbsolute = TypeReal (final unit="L3/N", min=0) "Absolute specific volume";</code>
<code>type VolumeSpecificRate = TypeReal (final unit="L3/(N.T)") "Rate of specific volume";</code>
<code>type Wavenumber = TypeReal (final unit="A/L");</code>
<code>type CapacityThermal = Amount (displayUnit="J/K") "Thermal capacity";</code>
<code>type CapacityThermalSpecific = NumberAbsolute (displayUnit="J/(mol.K)") "Specific thermal capacity";</code>
<code>type CapacityThermalVolumic = Density (displayUnit="J/(m3.K)") "Volumic thermal capacity";</code>

```
type PotentialChemical = Potential (displayUnit="J/mol") "Chemical
  potential";
```

```
type Temperature = Potential (displayUnit="K");
```

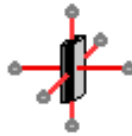
```
type TemperatureAbsolute = PotentialAbsolute (displayUnit="degC")
  "Absolute temperature";
```

```
type TemperatureRate = PotentialRate (displayUnit="K/s") "Rate of
  temperature";
```

```
type ResistivityThermal = Resistivity (displayUnit="m.K/W")
  "Thermal resistivity";
```

```
type Conductance = Current (displayUnit="W/K") "Conductance";
```

B.48 FCSys.Regions.AnCLs.AnCL



Anode catalyst layer

B.48.1 Information

This model represents the anode catalyst layer of a PEMFC. The x axis extends from the anode to the cathode. By default, the cross-sectional area in the yz plane is 50 cm².

The default thickness ($L_x = \{28.7 \cdot U \cdot \text{um}\}$) is from [94]. The default thermal conductivity of the carbon ($\theta = U \cdot \text{m} \cdot U \cdot \text{K} / (1.18 \cdot U \cdot \text{W})$) represents a compressed SGL Sigracet 10 BA gas diffusion layer [225]. The default electronic conductivity ($\sigma = 40 \cdot U \cdot \text{S} / (12 \cdot U \cdot \text{cm})$) is for SGL Carbon Group Sigracet[®] 10 BA (see AnGDLS.Sigracet10BA).

Default assumptions (may be adjusted):

1. All of the species have the same temperature, even in different phases.
2. Half of the solid is graphite and half is ionomer (by volume).

For more information, please see the Region model.

This class extends from Region (Base model for a 3D array of subregions).

B.48.2 Modelica definition

```

model AnCL "Anode catalyst layer"
import Modelica.Constants.inf;
// extends FCSys.Icons.Names.Top4;

extends Region(
  L_x={28.7}*U.um,
  L_y={8}*U.cm,
  L_z={6.25}*U.cm,
  final inclTransX=true,
  inclTransY=false,
  inclTransZ=false,
  redeclare replaceable model Subregion = Subregions.Subregion (
    common(k_Q=0),
    gasLiq(k_Q=inf),
    gas(
      common(k_Q=inf),
      k=fill(epsilon^(-0.5), 3),
      inclH2=true,
      inclH2O=true,
      H2(
        initEnergy=Init.none,
        upstreamX=false,
        final zeta=0,
        phi(each stateSelect=StateSelect.always, each fixed=true)),
      H2O(
        initEnergy=Init.none,
        upstreamX=false,
        final zeta=0)),
    graphite(
      k=fill((0.5*(1 - epsilon))^(-0.5), 3),
      'inclC+'=true,
      'inclC-'=true,
      'inclC-Transfer'=true,
      'C+'(theta=U.m*U.K/(1.18*U.W)*(0.5*(1 - epsilon))^1.5,
        epsilon=0.5*(1 -
          epsilon)),
      'e-'(sigma=40*U.S/(12*U.cm)/(0.5*(1 - epsilon))^1.5)),
    ionomer(
      k=fill((0.5*(1 - epsilon))^(-0.5), 3),
      'inclSO3-'=true,
      'inclH+'=true,
      inclH2O=true,
      'H+'(initEnergy=Init.none, sigma=0.1*U.S/U.cm),
      'SO3-'(epsilon=0.5*(1 - epsilon),T(each fixed=false, each
        stateSelect
          =StateSelect.default)),
      H2O(initEnergy=Init.none, phi(each stateSelect=StateSelect.always,
        each fixed=true))),
    liquid(
      k=fill(epsilon^(-0.5), 3),
      inclH2O=true,
      H2O(
        upstreamX=false,

```

```

        epsilon_IC=1e-5,
        phi(each stateSelect=StateSelect.always, each fixed=true),
        each initEnergy=Init.none,
        NO=0.1*U.C,
        T(each stateSelect=StateSelect.default))),
        volume(inclCapillary=true, capillary(R=5*U.um))),
        subregions(graphite('e-Transfer'(final IO=J0*subregions.A[Axis.x]))));

// See the documentation layer of Phases.PartialPhase regarding the
// settings of k for each phase.

parameter Q.NumberAbsolute epsilon(nominal=1) = 0.4 "Porosity";

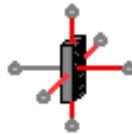
parameter Q.CurrentAreic J0(min=0) = U.A/U.cm^2
    "Exchange current density @ 300 K";

protected
    outer Conditions.Environment environment "Environmental conditions";

end AnCL;

```

B.49 FCSys.Regions.AnFPs.AnFP



Anode flow plate

B.49.1 Information

This model represents the anode flow plate of a PEMFC. The x axis extends from the anode to the cathode. Fluid is considered to travel in the y direction, with the associated length factor (k_y) greater than one (by default) to represent a serpentine channel. The model is bidirectional, meaning that either `yNegative` or `yPositive` can be used as the inlet. By default, the cross-sectional area in the yz plane is 50 cm².

The solid and the fluid phases are assumed to exist in the same subregions, even though a typical flow plate is impermeable to the fluid (except for the channel). In theory, it is possible to discretize the flow plate into smaller subregions for the bulk solid, lands, and valleys. However, this would significantly increase the mathematical size of the model. Currently, that level of detail is best left to computational fluid dynamics.

The x axis-component of the transport factor (k_x) for the gas and the liquid should generally be less than one because the transport distance into/out of the GDL is less than half the thickness of the flow plate. It is equal to the product of two ratios:

1. the depth of the channels to the thickness of the flow plate
2. the product of the total area of the flow plate in the yz plane (land + valleys) and the fraction of the total volume available for the fluid (ϵ) to the area of the valleys in the yz plane

See `Species.'C+'.Graphite.Fixed` regarding the default specific heat capacity. The default thermal resistivity of the carbon ($\theta = \text{U.m*U.K}/(95*\text{U.W})$) and the electrical conductivity ($\sigma = \text{U.S}/(1.470\text{e-}3*\text{U.cm})$) are that of Entegris/Poco Graphite AXF-5Q [228]. There is additional data in the text layer of this model.

For more information, please see the Region model.

This class extends from Region (Base model for a 3D array of subregions).

B.49.2 Modelica definition

```

model AnFP "Anode flow plate"
  import Modelica.Constants.inf;

  extends Region(
    L_x={10}*U.mm,
    L_y={8}*U.cm,
    L_z={6.25}*U.cm,
    final inclTransX=true,
    final inclTransY=true,
    inclTransZ=false,
    redeclare replaceable model Subregion =
      FCSys.Subregions.SubregionNoIonomer
      (
        common(k_Phi={1e7,inf,1e7},k_Q=1e5),
        gasLiq(k_Phi={inf,1e6,inf},k_Q=inf),
        gas(
          common(k_Phi={inf,inf,inf}),
          H2_H2O(k_Phi={inf,inf,inf}),
          k={0.22*epsilon,20.3/n_y,n_y/20.3},
          inclH2=true,
          inclH2O=true,
          H2(
            upstreamX=false,
            Nu_Phi={4,16*A[Axis.z]*epsilon/D^2,4},
            zeta=100*Characteristics.H2.Gas.zeta(),

```

```

    T(stateSelect=StateSelect.always)),
  H2O(
    upstreamX=false,
    Nu_Phi={4,16*A[Axis.z]*epsilon/D^2,4},
    zeta=100*Characteristics.H2O.Gas.zeta(),
    I(each stateSelect=StateSelect.always, each fixed=true),
    initEnergy=Init.none)),
  graphite(
    'inclC+'=true,
    'inclC-'=true,
    'C+'(theta=U.m*U.K/(95*U.W),epsilon=1 - epsilon),
    'e-'(sigma=U.S/(1.470e-3*U.cm))),
  liquid(
    k={0.22*epsilon,20.3/n_y,n_y/20.3},
    inclH2O=true,
    H2O(
      upstreamX=false,
      Nu_Phi={4,16*A[Axis.z]*epsilon/D^2,4},
      epsilon_IC=1e-5,
      NO=0.1*U.C)),
  volume(inclCapillary=false)));

parameter Q.NumberAbsolute epsilon(nominal=1) = 0.0588
  "Fraction of volume for the fluid";

parameter Q.Length D=0.937*U.mm "Hydraulic diameter of the channel";

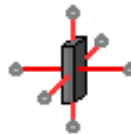
protected
Q.Velocity phi_states_H2[:, :, :](
  each stateSelect=StateSelect.always,
  each start=0,
  each fixed=true) = subregions[:, 2:n_y, :].gas.H2.phi[2] if n_y > 1
  "Forced states for H2";
// Note: This avoids dynamic state selection in Dymola 2014.

outer Conditions.Environment environment "Environmental conditions";

end AnFP;

```

B.50 FCSys.Regions.AnGDLs.AnGDL



Anode gas diffusion layer

B.50.1 Information

This model represents the anode gas diffusion layer of a PEMFC. The x axis extends from the anode to the cathode. By default, the cross-sectional area in the yz plane is 50 cm².

The default porosity ($\epsilon = 0.88$) is that of SGL Carbon Group Sigracet[®] 24 BC GDs. The porosity of a GDL may be lower than specified due to compression (e.g., 0.4 according to [67, p. 2483], although that reference may be outdated). The default thermal conductivity of the carbon ($\theta = U.m*U.K/(1.18*U.W)$) represents a compressed Sigracet[®] 10 BA gas diffusion layer [225]. The default electrical conductivity is also for Sigracet[®] 10 BA [221].

Default assumptions (may be adjusted):

1. All of the species have the same temperature, even in different phases.

For more information, please see the Region model.

This class extends from Region (Base model for a 3D array of subregions).

B.50.2 Modelica definition

```
model AnGDL "Anode gas diffusion layer"
import Modelica.Constants.inf;

extends Region(
  L_x={0.235}*U.mm,
  L_y={8}*U.cm,
  L_z={6.25}*U.cm,
  final inclTransX=true,
  inclTransY=false,
  inclTransZ=false,
  redeclare replaceable model Subregion =
    FCSys.Subregions.SubregionNoIonomer
    (
      common(k_Q=0),
      gasLiq(k_Q=inf),
      gas(
        common(k_Q=inf),
        k=fill(epsilon^(-0.5), 3),
        inclH2=true,
        inclH2O=true,
        H2(
          initEnergy=Init.none,
          upstreamX=false,
          phi(each stateSelect=StateSelect.always, each fixed=true),
          final zeta=0,
          final eta=0),
        H2O(
```

```

        initEnergy=Init.none,
        upstreamX=false,
        phi(each stateSelect=StateSelect.always, each fixed=true),
        final zeta=0,
        final eta=0)),
    graphite(
        k=fill((1 - epsilon)^(-0.5), 3),
        'inclC+'=true,
        'inclC-'=true,
        'C+'(theta=U.m*U.K/(1.18*U.W)*(1 - epsilon)^1.5,final epsilon=1 -
            epsilon),
        'e-'(sigma=40*U.S/(12*U.cm)/(1 - epsilon)^1.5)),
    liquid(
        k=fill(epsilon^(-0.5), 3),
        inclH2O=true,
        H2O(
            upstreamX=false,
            epsilon_IC=1e-5,
            final eta=0,
            phi(each stateSelect=StateSelect.always, each fixed=true),
            each initEnergy=Init.none,
            T(each stateSelect=StateSelect.default))),
        volume(kappa=1e-4*U.mm^2)));

// Note: The fluid species have zero fluidity (eta=0) so that the
// transverse
// velocity is zero at the interface with the flow plate. That condition
// is necessary to produce the appropriate pressure loss down the channel.

// See the documentation layer of Phases.PartialPhase regarding the
// settings of k for each phase.

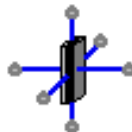
parameter Q.NumberAbsolute epsilon(nominal=1) = 0.8 "Porosity";

protected
    outer Conditions.Environment environment "Environmental conditions";

end AnGDL;

```

B.51 FCSys.Regions.CaCLs.CaCL



Cathode catalyst layer

B.51.1 Information

This model represents the cathode catalyst layer of a PEMFC. It is identical to the anode catalyst layer except for the included species, the exchange current density (J^0), and the activation energy (). For more information, please see the anCL model.

This class extends from Region (Base model for a 3D array of subregions).

B.51.2 Modelica definition

```
model CaCL "Cathode catalyst layer"
  import Modelica.Constants.inf;
  // extends FCSys.Icons.Names.Top4;

  extends Region(
    L_x={28.7}*U.um,
    L_y={8}*U.cm,
    L_z={6.25}*U.cm,
    final inclTransX=true,
    inclTransY=false,
    inclTransZ=false,
    redeclare replaceable model Subregion = Subregions.Subregion (
      common(k_Q=0),
      gasLiq(k_Q=inf),
      gas(
        common(k_Q=inf),
        k=fill(epsilon^(-0.5), 3),
        inclH2O=true,
        inclN2=true,
        inclO2=true,
        H2O(
          initEnergy=Init.none,
          upstreamX=false,
          phi(each stateSelect=StateSelect.always, each fixed=true),
          final zeta=0),
        N2(
          initEnergy=Init.none,
          upstreamX=false,
          final zeta=0),
        O2(
          initEnergy=Init.none,
          upstreamX=false,
          I(each stateSelect=StateSelect.always, each fixed=true),
          final zeta=0,
          p_stop=12*U.Pa)),
      graphite(
        k=fill((0.5*(1 - epsilon))^(-0.5), 3),
        'inclC+'=true,
        'inclC-'=true,
        'inclC-Transfer'=true,
```

```

'C+'(theta=U.m*U.K/(1.18*U.W)*(0.5*(1 - epsilon))^1.5,
      epsilon=0.5*(1 -
        epsilon)),
'e-'(sigma=40*U.S/(12*U.cm)/(0.5*(1 - epsilon))^1.5),
'e-Transfer'(E_A=0.75*U.V)),
ionomer(
  k=fill((0.5*(1 - epsilon))^(-0.5), 3),
  'inclS03-'=true,
  'inclH+'=true,
  inclH20=true,
  'S03-'(epsilon=0.5*(1 - epsilon), T(each fixed=false, each
    stateSelect=StateSelect.default)),
  'H+'(initEnergy=Init.none, sigma=0.1*U.S/U.cm),
  H20(initEnergy=Init.none, phi(each stateSelect=StateSelect.always,
    each fixed=true))),
liquid(
  k=fill(epsilon^(-0.5), 3),
  inclH20=true,
  H20(
    upstreamX=false,
    epsilon_IC=1e-5,
    phi(each stateSelect=StateSelect.always, each fixed=true),
    each initEnergy=Init.none,
    NO=0.1*U.C,
    T(each stateSelect=StateSelect.default))),
  volume(inclCapillary=true, capillary(R=5*U.um)),
  subregions(graphite('e-Transfer'(final IO=JO*subregions.A[Axis.x]))));

// See the documentation layer of Phases.PartialPhase regarding the
// settings of k for each phase.

parameter Q.NumberAbsolute epsilon(nominal=1) = 0.4 "Porosity";

parameter Q.CurrentAreic JO(min=0) = 0.023*U.mA/U.cm^2
  "Exchange current density @ 300 K";

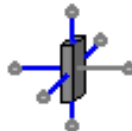
protected
  outer Conditions.Environment environment "Environmental conditions";

initial equation

end CaCL;

```

B.52 FCSys.Regions.CaFPs.CaFP



Cathode flow plate

B.52.1 Information

This model represents the cathode flow plate of a PEMFC. It is identical to the anode flow plate except for the included species, the default length of the channel, and the fraction of the volume for the fluid (ϵ). For more information, please see the anFP model.

This class extends from Region (Base model for a 3D array of subregions).

B.52.2 Modelica definition

```
model CaFP "Cathode flow plate"
  import Modelica.Constants.inf;

  extends Region(
    L_x={10}*U.mm,
    L_y={8}*U.cm,
    L_z={6.25}*U.cm,
    final inclTransX=true,
    final inclTransY=true,
    inclTransZ=false,
    redeclare replaceable model Subregion =
      FCSys.Subregions.SubregionNoIonomer
      (
        common(k_Phi={1e7, inf, 1e7}, k_Q=1e5),
        gas(
          common(k_Phi={inf, inf, inf}),
          H2O_N2(k_Phi={inf, inf, inf}),
          H2O_O2(k_Phi={inf, inf, inf}),
          N2_O2(k_Phi={inf, inf, inf}),
          k={0.17*epsilon, 13.3/n_y, n_y/13.3},
          inclH2O=true,
          inclN2=true,
          inclO2=true,
          H2O(
            upstreamX=false,
            Nu_Phi={4, 16*A[Axis.z]*epsilon/D^2, 4},
            final zeta=0,
            T(stateSelect=StateSelect.always)),
          N2(
            upstreamX=false,
            Nu_Phi={4, 16*A[Axis.z]*epsilon/D^2, 4},
            final zeta=0,
            initEnergy=Init.none,
            I(each stateSelect=StateSelect.always, each fixed=true)),
          O2(
            upstreamX=false,
            Nu_Phi={4, 16*A[Axis.z]*epsilon/D^2, 4},
            final zeta=0,
            initEnergy=Init.none)),
        graphite(
          'inclC+'=true,
```

```

        'incl-e-'=true,
        'C+'(theta=U.m*U.K/(95*U.W),epsilon=1 - epsilon),
        'e-'(sigma=U.S/(1.470e-3*U.cm)),
    liquid(
        k={0.17*epsilon,13.3/n_y,n_y/13.3},
        inclH2O=true,
        H2O(
            upstreamX=false,
            Nu_Phi={4,16*A[Axis.z]*epsilon/D^2,4},
            epsilon_IC=1e-5,
            NO=0.1*U.C)),
        volume(inclCapillary=false)));

parameter Q.NumberAbsolute epsilon(nominal=1) = 0.0423
    "Fraction of volume for the fluid";

parameter Q.Length D=0.815*U.mm "Hydraulic diameter of the channel";

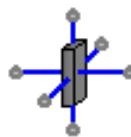
protected
Q.Velocity phi_states_H2O[:, :, :](
    each stateSelect=StateSelect.always,
    each start=0,
    each fixed=true) = subregions[:, 2:n_y, :].gas.H2O.phi[2] if n_y > 1
    "Forced states for H2O";
Q.Velocity phi_states_O2[:, :, :](
    each stateSelect=StateSelect.always,
    each start=0,
    each fixed=true) = subregions[:, 2:n_y, :].gas.O2.phi[2] if n_y > 1
    "Forced states for O2";
// Note: These variables avoid dynamic state selection in Dymola 2014.

outer Conditions.Environment environment "Environmental conditions";

end CaFP;

```

B.53 FCSys.Regions.CaGDLs.CaGDL



Cathode gas diffusion layer

B.53.1 Information

This model represents the cathode gas diffusion layer of a PEMFC. It is identical to the anode gas diffusion layer except for the included species. For more information, please see the anGDL model.

This class extends from Region (Base model for a 3D array of subregions).

B.53.2 Modelica definition

```
model CaGDL "Cathode gas diffusion layer"
  import Modelica.Constants.inf;

  extends Region(
    L_x={0.235}*U.mm,
    L_y={8}*U.cm,
    L_z={6.25}*U.cm,
    final inclTransX=true,
    inclTransY=false,
    inclTransZ=false,
    redeclare replaceable model Subregion =
      FCSys.Subregions.SubregionNoIonomer
      (
        common(k_Q=0),
        gasLiq(k_Q=inf),
        gas(
          common(k_Q=inf),
          k=fill(epsilon^(-0.5), 3),
          inclH2O=true,
          inclN2=true,
          inclO2=true,
          H2O(
            initEnergy=Init.none,
            upstreamX=false,
            phi(each stateSelect=StateSelect.always, each fixed=true),
            final zeta=0,
            final eta=0),
          N2(
            initEnergy=Init.none,
            upstreamX=false,
            phi(each stateSelect=StateSelect.always, each fixed=true),
            final zeta=0,
            final eta=0),
          O2(
            initEnergy=Init.none,
            upstreamX=false,
            I(each stateSelect=StateSelect.always, each fixed=true),
            final zeta=0,
            final eta=0)),
        graphite(
          k=fill((1 - epsilon)^(-0.5), 3),
          'inclC+'=true,
          'inclC-'=true,
          'C+'(theta=U.m*U.K/(1.18*U.W)*(1 - epsilon)^1.5, final epsilon=1 -
            epsilon),
          'e-'(sigma=40*U.S/(12*U.cm)/(1 - epsilon)^1.5)),
        liquid(
          k=fill(epsilon^(-0.5), 3),
          inclH2O=true,
          H2O(
            upstreamX=false,
            epsilon_IC=1e-5,
```

```

    final eta=0,
    phi(each stateSelect=StateSelect.always, each fixed=true),
    each initEnergy=Init.none,
    T(each stateSelect=StateSelect.default))),
    volume(inclCapillary=true,capillary(R=22*U.um)));

// Note: The fluid species have zero fluidity (eta=0) so that the
// transverse
// velocity is zero at the interface with the flow plate. That condition
// is necessary to produce the appropriate pressure loss down the channel.

// See the documentation layer of Phases.PartialPhase regarding the
// settings of k for each phase.

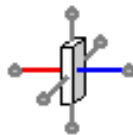
parameter Q.NumberAbsolute epsilon(nominal=1) = 0.8 "Porosity";

protected
    outer Conditions.Environment environment "Environmental conditions";

end CaGDL;

```

B.54 FCSys.Regions.PEMs.PEM



Proton exchange membrane

B.54.1 Information

This model represents the proton exchange membrane of a PEMFC. The x axis extends from the anode to the cathode. By default, the cross-sectional area in the yz plane is 50 cm².

The transport of protons includes inertance or inductance (i.e., translational momentum is stored) in the x direction in the subregions with index ($x = 1, y > 1, z > 1$). This is for numerical reasons, although in reality there is inductance.

Assumptions:

1. There are no pores in the PEM. All H₂O is absorbed into the ionomer itself.
2. There is no cross-over of the reactant gases.

For more information, please see the Region model.

This class extends from Region (Base model for a 3D array of subregions).

B.54.2 Modelica definition

```
model PEM "Proton exchange membrane"

  extends Region(
    L_x={40}*U.um,
    L_y={8}*U.cm,
    L_z={6.25}*U.cm,
    final inclTransX=true,
    inclTransY=false,
    inclTransZ=false,
    redeclare replaceable model Subregion =
      FCSys.Subregions.SubregionIonomer (
        ionomer(
          'inclSO3-'=true,
          'inclH+'=true,
          inclH2O=true,
          'SO3-'(final mu=0,final epsilon=1),
          'H+'(initEnergy=Init.none, sigma=0.1*U.S/U.cm),
          H2O(initEnergy=Init.none,upstreamX=false)),
        subregions(ionomer('H+'(consTransX={{if x > 1 or (y == 1 and z == 1)
          then
            ConsTrans.steady else ConsTrans.dynamic for z in 1:n_z} for y
              in 1
                :n_y} for x in 1:n_x}))))));

protected
  Q.Current I_states_1[:](
    each stateSelect=StateSelect.always,
    each start=0,
    each fixed=true) = subregions[1, 2:n_y, 1].ionomer.'H+'.I[1] if n_y > 1
    "Forced states, set #1";
  Q.Current I_states_2[:, :](
    each stateSelect=StateSelect.always,
    each start=0,
    each fixed=true) = subregions[1, :, 2:n_z].ionomer.'H+'.I[1] if n_z > 1
    "Forced states, set #2";
  // Note: These variables avoid dynamic state selection in Dymola 2014.

  outer Conditions.Environment environment "Environmental conditions";

end PEM;
```

B.55 FCSys.Regions.Region



Base model for a 3D array of subregions

B.55.1 Information

If L_x is an empty vector (e.g., `zeros(0)`, `ones(0)`, or `fill(1, 0)`), then there are no subregions along the x axis and the boundaries along the x axis are directly connected. The same applies to the other axes.

Table B.59: Parameters of FCSys.Region.Region.

Type	Name	Default	Description
replaceable model Subregion	Subregions.Subregion	Base subregion model	
		—Geometry—	
Length	L_x[:]	{U.cm}	Lengths along the x axis [L]
Length	L_y[:]	{U.cm}	Lengths along the y axis [L]
Length	L_z[:]	{U.cm}	Lengths across the z axis [L]
		—Assumptions—	
		—Included transport axes—	
Boolean	inclTransX	true	X
Boolean	inclTransY	true	Y
Boolean	inclTransZ	true	Z

Table B.59 lists the parameters of this class.

B.56 FCSys.Species.'C+'.Graphite.Fixed



Fixed properties

B.56.1 Information

Assumptions:

1. The thermal independity and thermal resistivity are fixed (e.g., independent of thermodynamic state).
2. The specific heat capacity is independent of temperature.
3. Mobility is zero.

The default isobaric specific heat capacity (via $b_c = [935 \cdot \text{U} \cdot \text{J} \cdot \text{Data} \cdot \text{m} / (\text{U} \cdot \text{kg} \cdot \text{U} \cdot \text{K})]$) and thermal resistivity ($\theta = \text{U} \cdot \text{m} \cdot \text{U} \cdot \text{K} / (11.1 \cdot \text{U} \cdot \text{W})$) are for graphite fiber epoxy (25% vol) composite (with heat flow parallel to the fibers) at 300 K [172, p. 909]. The integration offset for specific entropy is set such that the specific entropy is 154.663 J/(mol·K) at 25 °C and p° (1 atm). This is the value from Table B in [142].

For more information, please see the Species model.

This class extends from Solid (Base model for an inert, stationary solid).

B.56.2 Modelica definition

```
model Fixed "Fixed properties"

extends Solid(
  redeclare replaceable package Data = Characteristics.'C+'.Graphite (
    n_c=0,
    T_lim_c={0,Modelica.Constants.inf},
    b_c=[935*U.J*Data.m/(U.kg*U.K)],
    B_c=[Data.Deltah0_f - (935*U.J*Data.m/U.kg)*298.15,
        154.663*U.J/(U.mol*
            U.K) - Data.b_c[1, 1]*log(298.15*U.K)],
    final mu=0,
    redeclare parameter Q.TimeAbsolute nu=Data.nu(),
    redeclare parameter Q.ResistivityThermal theta=U.m*U.K/(11.1*U.W));
```

```

// Note: Parameter expressions (e.g., nu=Data.nu(environment.T)) are not
// used here since they render the parameters unadjustable after
// translation
// in Dymola 2014. This also applies to the other species.
end Fixed;

```

B.57 FCSys.Species.'e-'.Graphite.Fixed



Fixed properties

B.57.1 Information

Assumptions:

1. The fluidity is infinite. All friction is by translational exchange with the the substrate (C+).
2. The thermal resistivity is infinite. All of the thermal conductance is attributed to the substrate (C+).
3. The conductivity is mapped to the mobility of the electrons by assuming that the mobility of the substrate (C+) is zero.

If `constTransX`, `constTransY`, or `constTransZ` is `ConstTrans.dynamic` (the default is `ConstTrans.steady` instead), then internal inductance is included according to the relative permeability (μ^*).

For more information, please see the Species model.

This class extends from Ion (Base model for an ion).

B.57.2 Modelica definition

```

model Fixed "Fixed properties"
  extends Ion(
    redeclare final package Data = Characteristics.'e-'.Graphite,
    final Nu_Phi,
    final Nu_Q,
    final consRot,
    final upstreamX=false,

```

```

final upstreamY=false,
final upstreamZ=false,
final initMaterial=Init.none,
final initEnergy=Init.none,
final consTransX=ConsTrans.steady,
final consTransY=ConsTrans.steady,
final consTransZ=ConsTrans.steady,
final consEnergy=ConsThermo.steady,
final zeta=0,
final eta=Modelica.Constants.inf,
final theta=Modelica.Constants.inf,
final N_IC,
final p_IC,
final h_IC,
final V_IC,
final rho_IC,
final g_IC,
final T_IC,
final nu=1,
final tauprime={0},
final NO,
final n_chem=1,
final kL=fill(Data.d*mustar*N/4, n_trans));

parameter Q.NumberAbsolute mustar=1 "Relative permeability";

end Fixed;

```

B.58 FCSys.Species.'H+'.Ionomer.Fixed



Fixed properties

B.58.1 Information

Assumptions:

1. The generalized resistivities (η , θ) are fixed (e.g., independent of thermodynamic state).
2. The electrochemical reaction rate is governed by the electrons. Therefore, the specific exchange current is zero ($\tau' = 0$) for protons.
3. The conductivity is mapped to the mobility of the protons by assuming that the mobility of the substrate (e.g., C19HF3705S-) is zero.

The default electrical conductivity ($\sigma = 8.3 \cdot \text{U.S}/\text{U.m}$) is for DuPont™ Nafion® N-112 [229].

The default thermal resistivity ($\theta = \text{U.m} \cdot \text{U.K}/(0.1661 \cdot \text{U.W})$) is of H gas (rather than H^+) at 300 K from [175, p. 139].

For more information, please see the Species model.

This class extends from Ion (Base model for an ion).

B.58.2 Modelica definition

```
model Fixed "Fixed properties"
  extends Ion(
    redeclare replaceable package Data = Characteristics.'H+'.Ionomer,
    final Nu_Phi,
    final Nu_Q,
    final consRot,
    final upstreamX=false,
    final upstreamY=false,
    final upstreamZ=false,
    final initMaterial=Init.none,
    final consEnergy=ConsThermo.steady,
    consTransX=ConsTrans.steady,
    consTransY=ConsTrans.steady,
    consTransZ=ConsTrans.steady,
    final zeta=0,
    final N_IC,
    redeclare parameter Q.ResistivityThermal theta=U.m*U.K/(0.1661*U.W),
    final p_IC,
    final h_IC,
    final V_IC,
    final rho_IC,
    final g_IC,
    final T_IC,
    final nu=1,
    final tauprime,
    final NO,
    n_chem=1,
    sigma=8.3*U.S/U.m);
end Fixed;
```

B.59 FCSys.Species.'S03-'.Ionomer.Fixed



Fixed properties

B.59.1 Information

Assumptions:

1. The thermal independity and thermal resistivity are fixed (e.g., independent of thermodynamic state).

The default thermal resistivity ($\theta = U.m*U.K/(0.16*U.W)$) is of dry Nafion 115 [230, p. 1277].

For more information, please see the `Species` model.

This class extends from `Solid` (Base model for an inert, stationary solid).

B.59.2 Modelica definition

```
model Fixed "Fixed properties"
  extends Solid(
    redeclare replaceable package Data = Characteristics.'S03-' .Ionomer,
    redeclare parameter Q.Mobility mu=Modelica.Constants.small*U.C*U.s/U.g,
    redeclare parameter Q.TimeAbsolute nu=Data.nu(),
    redeclare parameter Q.ResistivityThermal theta=U.m*U.K/(0.16*U.W));

  // Note: Mobility is small but not zero to prevent a singularity when S03-
  // and C+ are both present (e.g., in the catalyst layers).

end Fixed;
```

B.60 FCSys.Species.Enumerations

Choices of options

B.60.1 Information

Table B.60 lists the contents of this class.

Table B.60: Contents of the `FCSys.Species.Enumerations` package.

Name	Description
Axis	Enumeration for Cartesian axes
Orient	Enumeration for orientations relative to a boundary
Side	Enumeration for sides of a region or subregion

continued on the next page ...

... continued from the previous page

Name	Description
ConsThermo	Options for the conservation of material or energy
ConsTrans	Options for the conservation of momentum
Init	Methods of initializing a thermodynamic quantity (material or energy)

B.60.2 Types and constants

```
type Axis = enumeration(  
  x "X",  
  y "Y",  
  z "Z") "Enumeration for Cartesian axes";
```

```
type Orient = enumeration(  
  after "Axis following the normal axis in Cartesian coordinates",  
  before "Axis preceding the normal axis in Cartesian coordinates")  
"Enumeration for orientations relative to a boundary";
```

```
type Side = enumeration(  
  n "Negative",  
  p "Positive (greater position along the Cartesian axis)")  
"Enumeration for sides of a region or subregion";
```

```
type ConsThermo = enumeration(  
  IC "Initial condition imposed forever (no conservation)",  
  steady "Steady (conservation with steady state)",  
  dynamic "Dynamic (conservation with storage)")  
"Options for the conservation of material or energy";
```

```
type ConsTrans = enumeration(  
  steady "Steady (conservation with steady state)",  
  dynamic "Dynamic (conservation with storage)")  
"Options for the conservation of momentum";
```

```
type Init = enumeration(  
  none "No initialization",  
  amount "Prescribed amount",  
  amountSS "Steady-state amount",  
  density "Prescribed density",  
  densitySS "Steady-state density",  
  volume "Prescribed volume",  
  volumeSS "Steady-state volume",  
  pressure "Prescribed pressure",  
  pressureSS "Steady-state pressure",  
  temperature "Prescribed temperature",  
  temperatureSS "Steady-state temperature",  
  specificEnthalpy "Prescribed specific enthalpy",
```

```
specificEnthalpySS "Steady-state specific enthalpy",
Gibbs "Prescribed Gibbs potential",
GibbsSS "Steady-state Gibbs potential")
"Methods of initializing a thermodynamic quantity (material or energy)";
```

B.61 FCSys.Species.Fluid



Base model for a fluid species

B.61.1 Information

Fixed assumptions:

1. The gradient of material current is uniform in the direction of the current.
2. The normal translational force on pairs of boundaries is split equally between the boundaries. This includes the body, shear (transverse translational transport), and exchange forces due to intermolecular drag and transfer during chemical reactions and phase change. It excludes the thermodynamic, dynamic (advective normal translational transport), and nonequilibrium (irreversible compression) pressures. It also excludes transient effects since translational momentum is stored at the boundaries (not in the subregion).
3. Nonequilibrium pressure is included in the thermodynamic states at the boundaries. In particular, the specific enthalpy at a boundary is a function of the temperature and the sum of the thermodynamic and nonequilibrium pressures at the boundary (and a possible artifact of dynamic pressure; see the first note regarding parameters). The rate of advection of energy is the product of this specific enthalpy and the material current.

Notes regarding the parameters:

1. If `approxVelocity` is `true`, then the normal velocities at the boundaries are calculated from the boundary currents assuming that the density is uniform. This avoids nonlinear systems of equations, but it introduces an artifact of the dynamic pressure into the thermodynamic states at the boundaries. The extra pressure is $m N_i^2 (v - v_i)/A'$, where m is the specific mass, v is the specific volume in the subregion, v_i is the specific volume at the

boundary, N_i is the boundary current, and A' is the available cross-sectional area. This affects the energy balance via the specific enthalpy at the boundaries.

2. If `consTransX`, `consTransY`, or `consTransZ` is `ConsTrans.steady`, then the derivative of translational momentum at and normal to the boundaries (proportional to $\partial N_i / \partial t$) is treated as zero and removed from the translational momentum balances/material transport equations at the corresponding boundaries.
3. If `consRot` is `true`, then rotational momentum is conserved without storage (i.e., steady). This means that the shear forces are mapped so that there is no net torque around any rotational axis that has all its boundaries included (i.e., all the boundaries around the perimeter). Rotational momentum is not exchanged among species or directly transported (i.e., uniform or shaft rotation).
4. Upstream discretization is applied by default. The central difference scheme may be used by setting `upstreamX`, `upstreamY`, and `upstreamZ` to `true`. The typical diffusion properties are such that the Péclet number for the upstream discretization of pressure will be much less (factor of 1/10,000) than the Péclet numbers for translational and thermal transport. Therefore, it may appear that pressure is not advected with the material transport stream.
5. The indices of the translational Nusselt number (Nu_ϕ) correspond to the orientation of the translational momentum that is transported, not the axes of material transport.
6. The default thermal Nusselt number is one, which represents pure conduction through the gas. Use 3.66 for internal flow where the boundaries are uniform in temperature or 48/11 (approximately 4.36) if the heat flux is uniform [172].

Translational momentum and thermal energy are advected as material is exchanged due to phase change and reactions. This occurs at the velocity (ϕ) and the specific entropy-temperature product (sT) of the reactants (source configurations), where the reactant/product designation depends on the current conditions.

The advective exchange is modeled via a `stream` connector (`Chemical`). The rate of advection of translational momentum is the product of the velocity of the source (ϕ) and the mass flow rate (M or mN). The rate of thermal advection is the specific entropy-temperature product

of the source (sT) times the rate of material exchange (N). If there are multiple sources, then their contributions are additive. If there are multiple sinks, then translational momentum is split on a mass basis and the thermal stream is split on a particle-number basis.

For more information, please see the `Species` model.

Table B.61 lists the parameters of this class.

This class extends from `Species` (Base model for one chemical species in one phase).

Table B.61: Parameters of `FCSys.Species.Fluid`.

Type	Name	Default	Description
Integer	<code>n_inter</code>	0	Number of exchange connections with other phases
replaceable package Data	<code>Characteristics.BaseClasses...</code>	—Material properties— Characteristic data	
Mobility	<code>mu</code>	<code>Data.mu(T, v)</code>	Mobility [N.T/M]
TimeAbsolute	<code>nu</code>	<code>Data.nu(T, v)</code>	Thermal independity [T]
Continuity	<code>zeta</code>	<code>Data.zeta(T, v)</code>	Continuity [L.M/(N.T)]
Fluidity	<code>eta</code>	<code>Data.eta(T, v)</code>	Fluidity [L.T/M]
ResistivityThermal	<code>theta</code>	<code>Data.theta(T, v)</code>	Thermal resistivity [L.T/N]
		—Independence factors—	
NumberAbsolute	<code>k_intra_Phi[n_intra, n_trans]</code>	<code>ones(n_intra, n_trans)</code>	For translational exchange among species within the phase [1]
NumberAbsolute	<code>k_intra_Q[n_intra]</code>	<code>ones(n_intra)</code>	For thermal exchange among species within the phase [1]
		—Initialization—	
TemperatureAbsolute	<code>T.start</code>	<code>T_IC</code>	Temperature [L2.M/(N.T2)]
Velocity	<code>phi.start[n_trans]</code>	0	Velocity [L/T]
		—Chemical parameters—	
TimeAbsolute	<code>tauprime[n_chem]</code>	<code>zeros(n_chem)</code>	Specific exchange currents [T]

continued on the next page . . .

... continued from the previous page

Type	Name	Default	Description
Length	kl[:]	—Geometry— L[cartTrans]	Effective transport length [L]
		—Initialization—	
Init	initMaterial	Init.pressure	Method of initializing the material state
Init	initEnergy	Init.temperature	Method of initializing the thermal state
Amount	N_IC		Initial amount of material [N]
Density	rho_IC		Initial density [N/L3]
Volume	V_IC		Initial volume [L3]
PressureAbsolute	p_IC		Initial pressure [M/(L.T2)]
TemperatureAbsolute	T_IC		Initial temperature [L2.M/(N.T2)]
Potential	h_IC		Initial specific enthalpy [L2.M/(N.T2)]
Potential	g_IC		Initial Gibbs potential [L2.M/(N.T2)]
		—Assumptions—	
Integer	n_trans	1	Number of transport axes
Integer	n_chem	0	Number of reaction and phase change processes
		—Formulation of the conservation equations—	
ConsThermo	consMaterial	ConsThermo.dynamic	Material
Boolean	consRot	false	Conserve rotational momentum

continued on the next page ...

... continued from the previous page

Type	Name	Default	Description
ConsTrans	consTransX	ConsTrans.dynamic	X-axis translational momentum
ConsTrans	consTransY	ConsTrans.dynamic	Y-axis translational momentum
ConsTrans	consTransZ	ConsTrans.dynamic	Z-axis translational momentum
ConsThermo	consEnergy	ConsThermo.dynamic	Energy
		—Axes with upstream discretization—	
Boolean	upstreamX	true	X
Boolean	upstreamY	true	Y
Boolean	upstreamZ	true	Z
		—Flow conditions—	
Boolean	approxVelocity	true	Calculate normal boundary velocities assuming uniform density
NumberAbsolute	Nu_Phi[Axis]	{4,4,4}	Translational Nusselt numbers [1]
NumberAbsolute	Nu_Q	1	Thermal Nusselt number [1]
		—Advanced—	
Amount	N0	0	Nominal amount of material to prevent depletion [N]

B.61.2 Modelica definition

```
model Fluid "Base model for a fluid species"

import FCSys.Utilities.Coordinates.after;
import FCSys.Utilities.Coordinates.before;
import FCSys.Utilities.Coordinates.cartWrap;
import FCSys.Utilities.Delta;
import FCSys.Utilities.Sigma;
import FCSys.Utilities.inSign;
import FCSys.Utilities.selectBooleans;
import FCSys.Utilities.selectIntegers;
import assert = FCSys.Utilities.assertEval;

// Initialization parameters
parameter Init initMaterial=Init.pressure
  "Method of initializing the material state";
parameter Init initEnergy=Init.temperature
  "Method of initializing the thermal state";
extends Species(N(stateSelect=if consMaterial == ConsThermo.dynamic then
  StateSelect.always else StateSelect.prefer),T(final fixed=false));
// Note: The extension is after these parameters so that they appear first
// in the parameter dialog.

// Material properties
parameter Integer n_chem=0 "Number of reaction and phase change processes";
Q.Continuity zeta(nominal=1e-3*U.N/U.A) = Data.zeta(T, v) "Continuity";
Q.Fluidity eta(nominal=1e5/(U.Pa*U.s)) = Data.eta(T, v) "Fluidity";
Q.ResistivityThermal theta(nominal=10*U.m*U.K/U.W) = Data.theta(T, v)
  "Thermal resistivity";

// Chemical parameters
Q.TimeAbsolute tauprime[n_chem](each nominal=U.ms) = zeros(n_chem)
  "Specific exchange currents";

// Geometry
Q.Length kL[:]=L[cartTrans] "Effective transport length";

// Assumptions
// -----
// Dynamics
parameter ConsThermo consMaterial=ConsThermo.dynamic "Material";
parameter Boolean consRot=false "Conserve rotational momentum";

parameter ConsTrans consTransX=ConsTrans.dynamic
  "X-axis translational momentum";
parameter ConsTrans consTransY=ConsTrans.dynamic
  "Y-axis translational momentum";
parameter ConsTrans consTransZ=ConsTrans.dynamic
  "Z-axis translational momentum";
parameter ConsThermo consEnergy=ConsThermo.dynamic "Energy";
//
// Upstream discretization
parameter Boolean upstreamX=true "X";
```

```

parameter Boolean upstreamY=true "Y";
parameter Boolean upstreamZ=true "Z";
//
// Flow conditions
parameter Boolean approxVelocity=true
  "Calculate normal boundary velocities assuming uniform density";
parameter Q.NumberAbsolute Nu_Phi[Axis]={4,4,4}
  "Translational Nusselt numbers";
parameter Q.NumberAbsolute Nu_Q=1 "Thermal Nusselt number";

// Advanced parameters
parameter Q.Amount NO=0 "Nominal amount of material to prevent depletion";

// Aliases (for common terms)
Q.Current I[n_trans](
  each nominal=U.A,
  each stateSelect=StateSelect.never,
  each start=0) "Current";
Q.Velocity phi_boundaries[n_trans, Side](
  each nominal=100*U.cm/U.s,
  each stateSelect=StateSelect.never,
  each start=0) "Normal velocities at the boundaries";
Q.Force f[n_trans](
  each nominal=U.N,
  each stateSelect=StateSelect.never,
  each start=0) "Total normal translational force on pairs of boundaries";
// This (f) includes the body, shear, and exchange forces due to
// intermolecular drag and transfer during chemical reactions and phase
// change. It excludes the thermodynamic, dynamic, and nonequilibrium
// compressive forces. It also excludes transient effects since
// translational momentum is stored at the boundaries.
Q.Force minusDeltaf[n_trans](
  each nominal=U.N,
  each stateSelect=StateSelect.never,
  each start=0) "Dynamic and nonequilibrium compression forces";

// Auxiliary variables (for analysis)
// -----
// Misc. conditions
output Q.Density rho_boundaries[n_trans, Side](each
  stateSelect=StateSelect.never)
  = fill(
    1,
    n_trans,
    2) ./ Data.v_Tp(boundaries.T, boundaries.p) if environment.analysis
  "Densities at the boundaries";
output Q.VolumeRate Vdot_boundaries[n_trans, Side](each stateSelect=
  StateSelect.never) = boundaries.Ndot ./ rho_boundaries if
  environment.analysis
  "Volume flow rates into the boundaries";
output Q.PressureAbsolute q[n_trans](each stateSelect=StateSelect.never) =
  (
    Data.m/2)*phi .* I ./ Aprime if environment.analysis "Dynamic pressure";

```

```

output Q.Velocity phi_chemical[n_chem, n_trans](each
    stateSelect=StateSelect.never)
    = actualStream(chemical.phi) if environment.analysis and n_chem > 0
    "Velocity of the chemical streams";
output Q.PotentialAbsolute sT_chemical[n_chem](each
    stateSelect=StateSelect.never)
    = actualStream(chemical.sT) if environment.analysis and n_chem > 0
    "Specific entropy-temperature product of the chemical streams";
output Q.Temperature DeltaT[n_trans](each stateSelect=StateSelect.never) =
    Delta(boundaries.T) if environment.analysis
    "Differences in temperatures across the boundaries";
output Q.Pressure Deltap[n_trans](each stateSelect=StateSelect.never) =
    Delta
    (boundaries.p) if environment.analysis
    "Differences in pressures across the boundaries";
output Q.Power Hprimedot[n_trans, Side](each stateSelect=StateSelect.never)
    = (Data.h(boundaries.T, boundaries.p) + Data.m*phi_boundaries .^ 2/2) .*
    boundaries.Ndot if environment.analysis
    "Flow rates of enthalpy + kinetic energy into the boundaries";
//
// Potentials
output Q.Potential g_boundaries[n_trans, Side](each
    stateSelect=StateSelect.never)
    = Data.g(boundaries.T, boundaries.p) if environment.analysis and not
    Data.isCompressible
    "Gibbs potentials at the boundaries";
output Q.Potential Deltag[n_trans](each stateSelect=StateSelect.never) =
    Delta(g_boundaries) if environment.analysis and not Data.isCompressible
    "Differences in Gibbs potentials across the boundaries";
// Note: If a boundary is left unconnected, then it's possible that its
// pressure may become negative. If the equation of state has an ideal-gas
// term, then the Gibbs energy will involve a logarithm of pressure.
// Therefore, to prevent errors, these variables are included only for
// incompressible species.
//
// Time constants
output Q.TimeAbsolute tau_NT[n_trans](
    each stateSelect=StateSelect.never,
    each start=U.s) = fill(zeta*beta*N, n_trans) ./ (2*Aprime) if
    environment.analysis
    "Time constants for material transport";
output Q.TimeAbsolute tau_PhiT[n_trans](
    each stateSelect=StateSelect.never,
    each start=U.s) = M*eta*kL ./ (2*Nu_Phi[cartTrans] .* Aprime) if
    environment.analysis
    "Time constants for transverse translational transport";
output Q.TimeAbsolute tau_QT[n_trans](
    each stateSelect=StateSelect.never,
    each start=U.s) = (N*c_v*theta/(2*Nu_Q))*kL ./ Aprime if
    environment.analysis
    "Time constants for thermal transport";
//
// Peclet numbers

```



```

output Q.Number Pe_N[n_trans](each stateSelect=StateSelect.never) = tau_NT
.*
I/N if environment.analysis "Material Peclet numbers";
output Q.Number Pe_Phi[n_trans](each stateSelect=StateSelect.never) =
tau_PhiT .* I/N if environment.analysis "Translational Peclet numbers";
output Q.Number Pe_Q[n_trans](each stateSelect=StateSelect.never) = tau_QT
.*
I/N if environment.analysis "Thermal Peclet numbers";
// Note: These Peclet numbers are calculated at the center of the
// subregion (for simplicity), but the Peclet numbers used in the transport
// equations are at each boundary.
//
// Bulk flow rates
output Q.Force mphil[n_trans, n_trans](each stateSelect=StateSelect.never)
=
outerProduct(I, Data.m*phi) if environment.analysis
"Bulk rate of translational advection (1st index: transport axis, 2nd
index: translational component)";
output Q.VolumeRate Vdot[n_trans](each stateSelect=StateSelect.never) = v*I
if environment.analysis "Bulk volumetric flow rate";
output Q.Power hI[n_trans](each stateSelect=StateSelect.never) = h*I if
environment.analysis "Bulk enthalpy flow rate";
//
// Translational momentum balance
output Q.Force Ma[n_trans](each stateSelect=StateSelect.never) =
M*(der(phi)/
U.s + environment.a[cartTrans]) + N*Data.z*environment.E[cartTrans] if
environment.analysis
"Acceleration force (including acceleration due to body forces)";
output Q.Force f_thermo[n_trans](each stateSelect=StateSelect.never) =
-Delta
(boundaries.p) .* Aprime if environment.analysis "Thermodynamic force";
output Q.Force f_AE[n_trans](each stateSelect=StateSelect.never) =
Data.m*sum
((actualStream(chemical[i].phi) - phi)*chemical[i].Ndot for i in
1:n_chem)
if environment.analysis "Acceleration force due to advective exchange";
output Q.Force f_AT[n_trans](each stateSelect=StateSelect.never) =
{sum((if
i == j then phi_boundaries[j, :] else boundaries[j, :].phi[cartWrap(
cartTrans[i] - cartTrans[j])]) - {phi[i], phi[i]})*boundaries[j, :].Ndot*
Data.m for j in 1:n_trans) for i in 1:n_trans} if environment.analysis
"Acceleration force due to advective transport";
output Q.Force f_DT[n_trans](each stateSelect=StateSelect.never) =
{sum(sum(
if i == j then {0,0} else boundaries[j, :].mPhidot[cartWrap(cartTrans[i]
-
cartTrans[j])]) for j in 1:n_trans) for i in 1:n_trans} if
environment.analysis
"Shear force from other subregions (diffusive transport)";
//
// Energy balance
output Q.Power Ndere(stateSelect=StateSelect.never) = (N*T*der(Data.s(T,
p)) +

```

```

M*phi*der(phi))/U.s if environment.analysis
"Rate of energy storage (internal and kinetic) and boundary work at
  constant mass";
// Note that T*der(s) = der(u) + p*der(v).
output Q.Power Edot_AE(stateSelect=StateSelect.never) = sum((chemical[i].g
+
  actualStream(chemical[i].sT) - h + (actualStream(chemical[i].phi)*
  actualStream(chemical[i].phi) - phi*phi)*Data.m/2)*chemical[i].Ndot for i
  in 1:n_chem) if environment.analysis
"Relative rate of energy (internal, flow, and kinetic) due to reactions
  and phase change";
output Q.Power Edot_AT(stateSelect=StateSelect.never) = sum((Data.h(
  boundaries[i, :].T, boundaries[i, :].p) - {h,h} + (phi_boundaries[i, :]
  .^ 2
  + sum(boundaries[i, :].phi[orient] .^ 2 for orient in Orient) -
  fill(phi*
  phi, 2))*(Data.m/2))*boundaries[i, :].Ndot for i in 1:n_trans) if
  environment.analysis
"Relative rate of energy (internal, flow, and kinetic) due to advective
  transport";
output Q.Power Edot_DT(stateSelect=StateSelect.never) =
  sum(sum(boundaries[i,
  :].phi[orient]*boundaries[i, :].mPhidot[orient] for orient in Orient)
  for i
  in 1:n_trans) + sum(boundaries.Qdot) if environment.analysis
"Rate of diffusion of energy from other subregions";
// Note: The structure of the problem shouldn't change if these
// auxiliary variables are included (hence, StateSelect.never).

Connectors.Boundary boundaries[n_trans, Side](
  each p(start=p_IC),
  each T(start=T_IC),
  Ndot(each start=0, each stateSelect=StateSelect.never))
"Connectors for transport";
Connectors.Chemical chemical[n_chem](
  each final n_trans=n_trans,
  g(each start=g_IC, each final fixed=false),
  sT(each start=h_IC - g_IC, each final fixed=false))
"Connector for reactions and phase change";

// Geometric parameters
protected
outer Q.Area Aprime[n_trans] "Effective cross-sectional area";
outer parameter Boolean inclRot[3]
  "true, if each axis of rotation has all its tangential boundaries
  included";
outer parameter Boolean inclTrans[3]
  "true, if each transport axis is included";
outer parameter Integer cartRot[:];
  "Cartesian-axis indices of the components of rotational momentum";
outer parameter Integer transCart[3]
  "Boundary-pair indices of the Cartesian axes";
final parameter ConsTrans consTrans[n_trans]=selectIntegers({consTransX,
  consTransY,consTransZ}, cartTrans)

```

```

    "Formulation of the translational conservation equations for the
      transport axes";
final parameter Boolean
  upstream[n_trans]=selectBooleans({upstreamX,upstreamY,
  upstreamZ}, cartTrans)
  "true, if each transport axis uses upstream discretization";

// Additional aliases (for common terms)
Q.Force mPhidot_boundaries[n_trans, Side, Orient](each nominal=U.N, each
  stateSelect=StateSelect.never) "Directly-calculated shear forces";

outer Conditions.Environment environment "Environmental conditions";

initial equation
  // Check the initial conditions.
  assert(V >= 0, "The volume of " + getInstanceName() + " is negative.
Check that the volumes of the other phases are set properly.");
  assert(initMaterial <> initEnergy or initMaterial == Init.none or
  consMaterial == ConsThermo.steady or consEnergy == ConsThermo.steady,
    "The initialization methods for material and energy must be
    different (unless none).");

  // Material
  if consMaterial == ConsThermo.IC then
    // Ensure that a condition is selected since the state is prescribed.
    assert(initMaterial <> Init.none, "The material state of " +
      getInstanceName() + " is prescribed, yet its condition is not defined.
Choose any condition besides none.");
    elseif consMaterial == ConsThermo.dynamic then
      // Initialize since there's a time-varying state.
      if initMaterial == Init.amount then
        N = N_IC;
      elseif initMaterial == Init.amountSS then
        der(N) = 0;
      elseif initMaterial == Init.density then
        assert(Data.isCompressible or Data.hasThermalExpansion,
          getInstanceName() + " is isochoric, yet its material initial
          condition is based on density.");
        1/v = rho_IC;
      elseif initMaterial == Init.densitySS then
        assert(Data.isCompressible or Data.hasThermalExpansion,
          getInstanceName() + " is isochoric, yet its material initial
          condition is based on density.");
        der(1/v) = 0;
      elseif initMaterial == Init.volume then
        V = V_IC;
      elseif initMaterial == Init.volumeSS then
        der(V) = 0;
      elseif initMaterial == Init.pressure then
        p = p_IC;
        assert(Data.isCompressible, getInstanceName() + " is incompressible,
          yet its material initial condition is based on pressure.");
      elseif initMaterial == Init.pressureSS then
        der(p) = 0;

```

```

    assert(Data.isCompressible, getInstanceName() + " is incompressible,
           yet its material initial condition is based on pressure.");
elseif initMaterial == Init.temperature then
    T = T_IC;
elseif initMaterial == Init.temperatureSS then
    der(T) = 0;
elseif initMaterial == Init.specificEnthalpy then
    h = h_IC;
elseif initMaterial == Init.specificEnthalpySS then
    der(h) = 0;
elseif initMaterial == Init.Gibbs then
    g = g_IC;
elseif initMaterial == Init.GibbsSS then
    der(g) = 0;
    // Else, there's no initial equation since
    // initMaterial == Init.none or
    // consMaterial == ConsThermo.steady.
end if;
end if;

// Energy
if consEnergy == ConsThermo.IC then
    // Ensure that a condition is selected since the state is prescribed.
    assert(initEnergy <> Init.none, "The energy state of " +
           getInstanceName() + " is prescribed, yet its condition is not
           defined.
Choose any condition besides none.");
elseif consEnergy == ConsThermo.dynamic then
    // Initialize since there's a time-varying state.
    if initEnergy == Init.amount then
        N = N_IC;
    elseif initEnergy == Init.amountSS then
        der(N) = 0;
    elseif initEnergy == Init.density then
        1/v = rho_IC;
        assert(Data.isCompressible or Data.hasThermalExpansion,
               getInstanceName() + " is isochoric, yet its thermal initial
               condition is based on density.");
    elseif initEnergy == Init.densitySS then
        der(1/v) = 0;
        assert(Data.isCompressible or Data.hasThermalExpansion,
               getInstanceName() + " is isochoric, yet its thermal initial
               condition is based on density.");
    elseif initEnergy == Init.volume then
        V = V_IC;
    elseif initEnergy == Init.volumeSS then
        der(V) = 0;
    elseif initEnergy == Init.pressure then
        p = p_IC;
        assert(Data.isCompressible, getInstanceName() + " is incompressible,
               yet its thermal initial condition is based on pressure.");
    elseif initEnergy == Init.pressureSS then
        der(p) = 0;

```

```

    assert(Data.isCompressible, getInstanceName() + " is incompressible,
           yet its thermal initial condition is based on pressure.");
elseif initEnergy == Init.temperature then
    T = T_IC;
elseif initEnergy == Init.temperatureSS then
    der(T) = 0;
elseif initEnergy == Init.specificEnthalpy then
    h = h_IC;
elseif initEnergy == Init.specificEnthalpySS then
    der(h) = 0;
elseif initEnergy == Init.Gibbs then
    g = g_IC;
elseif initEnergy == Init.GibbsSS then
    der(g) = 0;
    // Else, there's no initial equation since
    // initEnergy == Init.none or
    // consEnergy == ConsThermo.steady.
end if;
end if;

equation
// Aliases (only to clarify and simplify other equations)
v*I = Aprime .* phi "Current vs. velocity";
for i in 1:n_trans loop
    for side in Side loop
        (if approxVelocity then v else Data.v_Tp(boundaries[i, side].T,
          boundaries[i, side].p))*boundaries[i, side].Ndot =
            inSign(side)*Aprime[
                i]*phi_boundaries[i, side] "Current vs. velocity at the boundaries";
    end for;
end for;
minusDeltaf = I .* phi*Data.m + zeta*Sigma(boundaries.Ndot);

// Assumptions
2*I = -Delta(boundaries.Ndot) "Linear current profile (assumption #1)";

// Properties upon outflow due to reaction and phase change
chemical.phi = fill(phi, n_chem);
chemical.sT = fill(h - g, n_chem);

// Material exchange
for i in 1:n_chem loop
    if tauprime[i] > Modelica.Constants.small then
        tauprime[i]*chemical[i].Ndot = (N + NO)*exp((chemical[i].g - g)/T) - N;
    else
        chemical[i].g = g;
    end if;
end for;

// Transport
for i in 1:n_trans loop
    for side in Side loop
        // Material

```

```

(if consTrans[i] == ConsTrans.dynamic then
  kL[i]*Data.m*der(boundaries[i,
side].Ndot)/U.s else 0) = (Aprime[i]*(boundaries[i, side].p - p) +
inSign(side)*phi_boundaries[i, side]*boundaries[i, side].Ndot*Data.m
-
minusDeltaf[i])*(if upstream[i] then 1 + exp(-zeta*Data.beta(T, p)*
boundaries[i, side].Ndot/(2*Aprime[i])) else 2) + inSign(side)*f[i];

// Translational momentum
kL[i]*eta*mPhidot_boundaries[i, side, Orient.after] = Aprime[i]*Nu_Phi[
after(cartTrans[i])]*(boundaries[i, side].phi[Orient.after] - (if
inclTrans[after(cartTrans[i])] then
  phi[transCart[after(cartTrans[i])]]
else 0))*(if upstream[i] then 1 +
  exp(-kL[i]*eta*Data.m*boundaries[i,
side].Ndot/(2*Aprime[i]*Nu_Phi[after(cartTrans[i])])) else 2)
"1st transverse";
kL[i]*eta*mPhidot_boundaries[i, side, Orient.before] =
Aprime[i]*Nu_Phi[
before(cartTrans[i])]*(boundaries[i, side].phi[Orient.before] - (if
inclTrans[before(cartTrans[i])] then
  phi[transCart[before(cartTrans[i])]]
else 0))*(if upstream[i] then 1 +
  exp(-kL[i]*eta*Data.m*boundaries[i,
side].Ndot/(2*Aprime[i]*Nu_Phi[before(cartTrans[i])])) else 2)
"2nd transverse";

// Thermal energy
kL[i]*theta*boundaries[i, side].Qdot = Aprime[i]*Nu_Q*(boundaries[i,
side].T
- T)*(if upstream[i] then 1 + exp(-kL[i]*theta*Data.c_v(T, p)*
boundaries[i, side].Ndot/(2*Aprime[i]*Nu_Q)) else 2);
end for;

// Direct mapping of shear forces (calculated above)
if not (consRot and inclRot[before(cartTrans[i])]) then
  boundaries[i, :].mPhidot[Orient.after] = mPhidot_boundaries[i, :,
Orient.after];
// Else, the force must be mapped for zero torque (below).
end if;
if not (consRot and inclRot[after(cartTrans[i])]) then
  boundaries[i, :].mPhidot[Orient.before] = mPhidot_boundaries[i, :,
Orient.before];
// Else, the force must be mapped for zero torque (below).
end if;
end for;

// Zero-torque mapping of shear forces
if consRot then
  for axis in cartRot loop
    4*cat(
      1,
      boundaries[transCart[after(axis)], :].mPhidot[Orient.after],

```

```

        boundaries[transCart[before(axis)], :].mPhidot[Orient.before]) =
            {{3,1,
L[before(axis)]/L[after(axis)],-L[before(axis)]/L[after(axis)]},{1,3,-L[
before(axis)]/L[after(axis)],L[before(axis)]/L[after(axis)]},{L[after(
axis)]/L[before(axis)],-L[after(axis)]/L[before(axis)],3,1},{-L[after(
axis)]/L[before(axis)],L[after(axis)]/L[before(axis)],1,3}}*cat(
1,
mPhidot_boundaries[transCart[after(axis)], :, Orient.after],
mPhidot_boundaries[transCart[before(axis)], :, Orient.before]);
    end for;
end if;

// Material dynamics
if consMaterial == ConsThermo.IC then
    // Apply the IC forever (material not conserved).
    if initMaterial == Init.amount then
        N = N_IC;
    elseif initMaterial == Init.amountSS then
        der(N) = 0;
    elseif initMaterial == Init.density then
        1/v = rho_IC;
    elseif initMaterial == Init.densitySS then
        der(1/v) = 0;
    elseif initMaterial == Init.volume then
        V = V_IC;
    elseif initMaterial == Init.volumeSS then
        der(V) = 0;
    elseif initMaterial == Init.pressure then
        p = p_IC;
    elseif initMaterial == Init.pressureSS then
        der(p) = 0;
    elseif initMaterial == Init.temperature then
        T = T_IC;
    elseif initMaterial == Init.temperatureSS then
        der(T) = 0;
    elseif initMaterial == Init.specificEnthalpy then
        h = h_IC;
    elseif initMaterial == Init.specificEnthalpySS then
        der(h) = 0;
    elseif initMaterial == Init.Gibbs then
        g = g_IC;
    else
        // if initMaterial == Init.GibbsSS then
        der(g) = 0;
        // Note: initMaterial == Init.none can't occur due to an assertion.
    end if;
else
    (if consMaterial == ConsThermo.dynamic then der(N)/U.s else 0) = sum(
        chemical.Ndot) + sum(boundaries.Ndot) "Material conservation";
end if;

// Conservation of translational momentum
f + M*environment.a[cartTrans] + Data.z*N*environment.E[cartTrans] =
    Data.m*

```

```

sum(actualStream(chemical[i].phi)*chemical[i].Ndot for i in 1:n_chem) + {
sum(intra[:].mPhidot[j]) + sum(inter[:].mPhidot[j]) + sum((if i == j
    then 0
    else boundaries[i, :].phi[cartWrap(cartTrans[j] - cartTrans[i])])*
boundaries[i, :].Ndot*Data.m + sum(boundaries[i, :].mPhidot[cartWrap(
    cartTrans[j] - cartTrans[i])))) for i in 1:n_trans) for j in 1:n_trans};
// Note: The storage is split between the boundaries via f, so a
// derivative doesn't appear here (see material transport above).

// Thermal dynamics
if consEnergy == ConsThermo.IC then
    // Apply the IC forever (energy not conserved).
    if initEnergy == Init.amount then
        N = N_IC;
    elseif initEnergy == Init.amountSS then
        der(N) = 0;
    elseif initEnergy == Init.density then
        1/v = rho_IC;
    elseif initEnergy == Init.densitySS then
        der(1/v) = 0;
    elseif initEnergy == Init.volume then
        V = V_IC;
    elseif initEnergy == Init.volumeSS then
        der(V) = 0;
    elseif initEnergy == Init.pressure then
        p = p_IC;
    elseif initEnergy == Init.pressureSS then
        der(p) = 0;
    elseif initEnergy == Init.temperature then
        T = T_IC;
    elseif initEnergy == Init.temperatureSS then
        der(T) = 0;
    elseif initEnergy == Init.specificEnthalpy then
        h = h_IC;
    elseif initEnergy == Init.specificEnthalpySS then
        der(h) = 0;
    elseif initEnergy == Init.Gibbs then
        g = g_IC;
    else
        // if initEnergy == Init.GibbsSS then
        der(g) = 0;
        // Note: initEnergy == Init.none can't occur due to an assertion.
    end if;
else
    (if consEnergy == ConsThermo.dynamic then (N*T*der(s) +
        der(M*phi*phi)/2)/U.s
    else 0) = sum((chemical[i].g + actualStream(chemical[i].sT) - h +
actualStream(chemical[i].phi)*actualStream(chemical[i].phi)*Data.m/2)*
chemical[i].Ndot for i in 1:n_chem) + sum(intra[i].phi*intra[i].mPhidot
for i in 1:n_intra) + sum(inter[i].phi*inter[i].mPhidot for i in 1:
n_inter) + sum(intra.Qdot) + sum(inter.Qdot) +
        sum((Data.h(boundaries[i,
:],T, boundaries[i, :].p) - {h,h} + (phi_boundaries[i, :] .^ 2 + sum(
boundaries[i, :].phi[orient] .^ 2 for orient in Orient))*(Data.m/2))*

```



```

    boundaries[i, :].Ndot + sum(boundaries[i, :].phi[orient]*boundaries[i,
    :].mPhidot[
    orient] for orient in Orient) for i in 1:n_trans) +
    sum(boundaries.Qdot)
    "Conservation of energy";
  end if;
end Fluid;

```

B.62 FCSys.Species.H2.Gas.Fixed



Fixed properties

B.62.1 Information

Assumptions:

1. The generalized resistivities (η , θ) are fixed (e.g., independent of thermodynamic state).
2. Ideal gas

The default resistivities ($\eta = 1/(89.6e-7*U.Pa*U.s)$ and $\theta = U.m*U.K/(183e-3*U.W)$) are based on data of H₂ gas at 1 atm and 300 K from Incropera and DeWitt [172, pp. 919–920].

The specific heat capacity is not fixed because it would affect the chemical potential and result in an incorrect cell potential.

For more information, please see the Species model.

This class extends from Fluid (Base model for a fluid species).

B.62.2 Modelica definition

```

model Fixed "Fixed properties"
  extends Fluid(
    redeclare replaceable package Data = FCSys.Characteristics.H2.Gas
      (b_v=[1],
       n_v={-1,0}),
    redeclare parameter Q.Mobility mu=Data.mu(),
    redeclare parameter Q.TimeAbsolute nu=Data.nu(),
    redeclare parameter Q.Continuity zeta=Data.zeta(),
    redeclare parameter Q.Fluidity eta=1/(8.96e-6*U.Pa*U.s),
    redeclare parameter Q.ResistivityThermal theta=U.m*U.K/(0.183*U.W),

```

```

final tauprime,
final NO,
n_chem=1,
p_IC=environment.p_dry);
end Fixed;

```

B.63 FCSys.Species.H2O.Gas.Fixed



Fixed properties

B.63.1 Information

Assumptions:

1. The generalized resistivities (η , θ) are fixed (e.g., independent of thermodynamic state).
2. Ideal gas
3. The specific exchange currents (τ') are zero. The rate of phase change is governed by the other configurations (liquid and ionomer).

The default resistivities ($\eta = 1/(9.09e-6 \cdot U.Pa \cdot U.s)$ and $\theta = U.m \cdot U.K/(19.6e-3 \cdot U-.W)$) are of H₂O gas at saturation pressure and 300 K from Incropera and DeWitt [172, p. 921].

The specific heat capacity is not fixed because it would affect the chemical potential and result in an incorrect cell potential.

The relative humidities (RH and RH_{boundaries}), which are calculated as output variables, do not account for surface tension.

For more information, please see the Species model.

This class extends from Fluid (Base model for a fluid species).

B.63.2 Modelica definition

```

model Fixed "Fixed properties"
  extends Fluid(

```

```

redeclare replaceable package Data = FCSys.Characteristics.H2O.Gas
  (b_v=[1],
   n_v={-1,0}),
redeclare parameter Q.Mobility mu=Data.mu(),
redeclare parameter Q.TimeAbsolute nu=Data.nu(),
redeclare parameter Q.Continuity zeta=Data.zeta(),
redeclare parameter Q.Fluidity eta=1/(9.09e-6*U.Pa*U.s),
redeclare parameter Q.ResistivityThermal theta=U.m*U.K/(19.6e-3*U.W),
final tauprime,
final NO,
n_chem=2,
p_IC=environment.p_H2O);

output Q.NumberAbsolute RH(
  stateSelect=StateSelect.never,
  displayUnit="%") = p/Characteristics.H2O.p_sat(T) if environment.analysis
  "Relative humidity (approximate)";
output Q.NumberAbsolute RH_boundaries[n_trans, Side](
  each stateSelect=StateSelect.never,
  each displayUnit="%") = boundaries.p ./ Characteristics.H2O.p_sat(
  boundaries.T) if environment.analysis
  "Relative humidity at the boundaries (approximate)";

end Fixed;

```

B.64 FCSys.Species.H2O.Ionomer.Fixed



Fixed properties

B.64.1 Information

Assumptions:

1. The generalized resistivities (η , θ) are fixed (e.g., independent of thermodynamic state).

For more information, please see the Species model.

This class extends from Fluid (Base model for a fluid species).

B.64.2 Modelica definition

```

model Fixed "Fixed properties"
  extends Fluid(
    redeclare replaceable package Data = Characteristics.H2O.Ionomer,
    redeclare parameter Q.Mobility mu=Data.mu(),

```

```

redeclare parameter Q.TimeAbsolute nu=Data.nu(),
redeclare parameter Q.Fluidity eta=Data.eta(),
redeclare parameter Q.ResistivityThermal theta=Data.theta(),
final N_IC,
final h_IC,
final g_IC,
final rho_IC,
final p_IC,
final V_IC,
final T_IC,
final consMaterial,
final tauprime,
n_chem=1,
final initMaterial=Init.none);

parameter Q.NumberAbsolute lambda_IC=14*environment.RH
  "Initial ratio of H2O molecules to SO3- end-groups";

Q.NumberAbsolute lambda(start=lambda_IC, fixed=true)
  "Ratio of H2O molecules to SO3- end-groups";

equation
  lambda*v = Characteristics.'SO3-'.Ionomer.b_v[1, 1];
end Fixed;

```

B.65 FCSys.Species.H2O.Liquid.Fixed



Fixed properties

B.65.1 Information

Assumptions:

1. The generalized resistivities (η , θ) are fixed (e.g., independent of thermodynamic state).

The default resistivities ($\eta = 1/(855e-6*U.Pa*U.s)$ and $\theta = U.m*U.K/(613e-3*U.W)$) are of H₂O liquid at saturation pressure and 300 K from Incropera and DeWitt [172, p. 921].

The specific heat capacity is not fixed because it would affect the chemical potential and result in an incorrect saturation pressure.

For more information, please see the Species model.

This class extends from Fluid (Base model for a fluid species).

B.65.2 Modelica definition

```
model Fixed "Fixed properties"

// Initialization
parameter Q.Number epsilon_IC=0.01 "Initial volumetric fill fraction";

extends Fluid(
  redeclare replaceable package Data = Characteristics.H2O.Liquid,
  redeclare parameter Q.Mobility mu=Data.mu(),
  redeclare parameter Q.TimeAbsolute nu=Data.nu(),
  final zeta=0,
  redeclare parameter Q.Fluidity eta=1/(855e-6*U.Pa*U.s),
  redeclare parameter Q.ResistivityThermal theta=U.m*U.K/(0.613*U.W),
  initEnergy=Init.temperature,
  final N_IC,
  final h_IC,
  final g_IC,
  final rho_IC,
  final p_IC,
  redeclare parameter Q.TimeAbsolute
    tauprime[:]={0,1e8,2e11}*Data.tauprime(),
  n_chem=3,
  final V_IC=epsilon_IC*product(L),
  initMaterial=Init.volume);

end Fixed;
```

B.66 FCSys.Species.Ion



Base model for an ion

B.66.1 Information

Please see the Fluid model.

This class extends from Fluid (Base model for a fluid species).

B.66.2 Modelica definition

```
model Ion "Base model for an ion"
  import assert = FCSys.Utilities.assertEval;
  extends Fluid(final mu=sigma*v, N(stateSelect=StateSelect.default));

  parameter Q.ConductivityElectrical sigma=Data.mu()/Data.v_Tp()
    "Electrical conductivity";
```

```
end Ion;
```

B.67 FCSys.Species.N2.Gas.Fixed



Fixed properties

B.67.1 Information

Assumptions:

1. The generalized resistivities (η , θ) are fixed (e.g., independent of thermodynamic state).
2. Ideal gas
3. Fixed specific heat capacity (independent of temperature)

The default specific heat capacity (via $b_c = [1.041e3*U.J*Data.m/(U.kg*U.K)]$) and resistivities ($\eta = 1/(17.82e-6*U.Pa*U.s)$ and $\theta = U.m*U.K/(25.9e-3*U.W)$) are based on data of gas at 1 atm and 300 K from Incropera and DeWitt [172, p. 920]. The integration offset for specific entropy (B_c) is set such that the specific entropy is 191.610 J/(mol·K) at 25 °C and p° (1 bar). This is the value from Table B in [142].

For more information, please see the Species model.

This class extends from Fluid (Base model for a fluid species).

B.67.2 Modelica definition

```
model Fixed "Fixed properties"
  import FCSys.Utilities.Polynomial;

  extends Fluid(
    redeclare replaceable package Data = FCSys.Characteristics.N2.Gas (
      b_v=[1],
      n_v={-1,0},
      n_c=0,
      T_lim_c={0,Modelica.Constants.inf},
      b_c=[1041*U.J*Data.m/(U.kg*U.K)],
      B_c=[Data.Deltah0_f - (1041*U.J*Data.m/U.kg)*298.15,
          191.610*U.J/(U.mol*
            U.K) - (1041*U.J*Data.m/(U.kg*U.K))*log(298.15*U.K)],
    redeclare parameter Q.Mobility mu=Data.mu(),
    redeclare parameter Q.TimeAbsolute nu=Data.nu(),
```

```

redeclare parameter Q.Continuity zeta=Data.zeta(),
redeclare parameter Q.Fluidity eta=1/(17.82e-6*U.Pa*U.s),
redeclare parameter Q.ResistivityThermal theta=U.m*U.K/(25.9e-3*U.W),
final tauprime,
final NO,
n_chem=0,
p_IC=environment.p_dry - environment.p_O2);
end Fixed;

```

B.68 FCSys.Species.O2.Gas.Fixed



Fixed properties

B.68.1 Information

Assumptions:

1. The generalized resistivities (η , θ) are fixed (e.g., independent of thermodynamic state).
2. Ideal gas

The default resistivities ($\eta = 1/(207.2e-7*U.Pa*U.s)$ and $\theta = U.m*U.K/(26.8e-3*U.W)$) are based on data of gas at 1 atm and 300 K from Incropera and DeWitt [172, pp. 920–921].

The specific heat capacity is not fixed because it would affect the chemical potential and result in an incorrect cell potential.

For more information, please see the Species model.

This class extends from Fluid (Base model for a fluid species).

B.68.2 Modelica definition

```

model Fixed "Fixed properties"
  extends Fluid(
    redeclare replaceable package Data = FCSys.Characteristics.O2.Gas
      (b_v=[1],
       n_v={-1,0}),
    redeclare parameter Q.Mobility mu=Data.mu(),
    redeclare parameter Q.TimeAbsolute nu=Data.nu(),

```

```

redeclare parameter Q.Continuity zeta=Data.zeta(),
redeclare parameter Q.Fluidity eta=1/(20.72e-6*U.Pa*U.s),
redeclare parameter Q.ResistivityThermal theta=U.m*U.K/(26.8e-3*U.W),
final tauprime,
final NO,
n_chem=1,
p_IC=environment.p_02);

parameter Q.Pressure p_stop=-Modelica.Constants.inf
  "Pressure below which the simulation should terminate";

equation
  if p_stop > 0 then
    when p < p_stop then
      terminate("There is no more " + Data.formula + ".");
    end when;
  end if;

end Fixed;

```

B.69 FCSys.Species.Solid



Base model for an inert, stationary solid

B.69.1 Information

Assumptions:

1. There is no material transport.
2. Velocity is zero.
3. There are no chemical reactions or phase change.
4. The volume is constant—determined by ϵ and the total volume of the subregion.

For more information, please see the `Species` model.

Table B.62 lists the parameters of this class. Table B.63 lists its connectors or connector variables.

This class extends from `Species` (Base model for one chemical species in one phase).

Table B.62: Parameters of `FCSys.Species.Solid`.

Type	Name	Default	Description
Integer	<code>n_inter</code>	0	Number of exchange connections with other phases
NumberAbsolute	<code>epsilon</code>	0.25	Volumetric fill fraction [1]
Length	<code>kL[:]</code>	<code>L[cartTrans]</code>	Effective transport length [L]
replaceable package Data	<code>Characteristics.BaseClasses...</code>	—Material properties— <code>Characteristic data</code>	
Mobility	<code>mu</code>	<code>Data.mu(T, v)</code>	Mobility [N.T/M]
TimeAbsolute	<code>nu</code>	<code>Data.nu(T, v)</code>	Thermal independency [T]
ResistivityThermal	<code>theta</code>	<code>Data.theta(T, v)</code>	Thermal resistivity [L.T/N]
NumberAbsolute	<code>k_intra_Phi[n_intra, n_trans]</code>	—Independence factors— <code>ones(n_intra, n_trans)</code>	For translational exchange among species within the phase [1]
NumberAbsolute	<code>k_intra_Q[n_intra]</code>	<code>ones(n_intra)</code>	For thermal exchange among species within the phase [1]
TemperatureAbsolute	<code>T.start</code>	—Initialization— <code>T_IC</code>	Temperature [L2.M/(N.T2)]
Velocity	<code>phi.start[n_trans]</code>	0	Velocity [L/T]
		—Assumptions—	

continued on the next page . . .

... continued from the previous page

Type	Name	Default	Description
Integer	n_trans	1	Number of transport axes
ConsThermo	consEnergy	ConsThermo.dynamic	Energy
		— Initialization —	
		1/Data.v_Tp0	Initial density [N/L ³]
Density	rho_IC	epsilon*product(L)	Initial volume [L ³]
Volume	V_IC	environment.p	Initial pressure [M/(L.T ²)]
PressureAbsolute	p_IC		Initial temperature [L ² .M/(N.T ²)]
TemperatureAbsolute	T_IC		

Table B.63: Connectors of FCSys.Species.Solid.

Type	Name	Description
Intra	intra[n_intra]	Connectors to exchange translational momentum and energy within the phase
Inter	inter[n_inter]	Connectors to exchange translational momentum and energy with all other species
Dalton	dalton	Connector for additivity of pressure
ThermalDiffusive	boundaries[n_trans, Side]	Connectors for transport

B.69.2 Modelica definition

```

model Solid "Base model for an inert, stationary solid"
  import FCSys.Utilities.Delta;

  // Geometry
  parameter Q.NumberAbsolute epsilon=0.25 "Volumetric fill fraction";
  extends Species(
    final N_IC,
    final V_IC=epsilon*product(L),
    final rho_IC=1/Data.v_Tp(),
    final p_IC=environment.p,
    final h_IC,
    final g_IC,
    T(stateSelect=if consEnergy == ConsThermo.dynamic then StateSelect.always
      else StateSelect.default, fixed=consEnergy == ConsThermo.dynamic));
  parameter Q.Length kL[:]=L[cartTrans] "Effective transport length";

  // Material properties
  Q.ResistivityThermal theta(nominal=10*U.cm/U.A) = Data.theta(T, v)
    "Thermal resistivity";

  // Assumptions
  parameter ConsThermo consEnergy=ConsThermo.dynamic "Energy";

  // Auxiliary variables (for analysis)
  output Q.TimeAbsolute tau_QT[n_trans](
    each stateSelect=StateSelect.never,
    each start=U.s) = N*c_v*theta*kL ./ (2*Aprime) if environment.analysis
    "Time constants for thermal transport (through the whole subregion)";

  output Q.Temperature DeltaT[n_trans](each stateSelect=StateSelect.never) =
    Delta(boundaries.T) if environment.analysis
    "Differences in temperatures across the boundaries";

```

```

Connectors.ThermalDiffusive boundaries[n_trans, Side](T(each start=T_IC))
  "Connectors for transport";

protected
  outer Q.Area Aprime[n_trans] "Effective cross-sectional area";

equation
  // Assumptions
  phi = zeros(n_trans) "Zero velocity";
  V = epsilon*product(L) "Prescribed volume";

  // Thermal transport (conduction)
  for i in 1:n_trans loop
    for side in Side loop
      kL[i]*theta*boundaries[i, side].Qdot = 2*Aprime[i]*(boundaries[i,
        side].T -
        T);
    end for;
  end for;

  // Thermal dynamics
  if consEnergy == ConsThermo.IC then
    // Apply the IC forever (energy not conserved).
    T = T_IC;
  else
    (if consEnergy == ConsThermo.dynamic then N*T*der(s)/U.s else 0) = sum(
      intra[i].phi*intra[i].mPhidot for i in 1:n_intra) + sum(inter[i].phi*
      inter[i].mPhidot for i in 1:n_inter) + sum(intra.Qdot) +
      sum(inter.Qdot) +
      sum(boundaries.Qdot) "Conservation of energy";
  end if;
end Solid;

```

B.70 FCSys.Species.Species



Base model for one chemical species in one phase

B.70.1 Information

All of the details below are pertinent to the Fluid model (and the derived Ion model) which inherits from this model. Only some of the details apply to the Solid model because it excludes the transport and exchange of material.

This model is based on the following fixed assumptions:

1. All boundaries are rectangular.
2. The material is orthorhombic. This implies that a gradient which induces diffusion along an axis does not induce diffusion along axes orthogonal to it [151, pp. 691–692].
3. The coordinate system (x, y, z) is aligned with the principle axes of transport. For example, if the material is stratified, then the layers must be parallel to one of the planes in the rectilinear grid.
4. The effective transport lengths (kL) are common to material, translational, and thermal transport.
5. There is no radiative heat transfer (or it must be linearized and added to the thermal conductance).
6. Rotational momentum is not exchanged, transported, or stored.

Other assumptions are optional via the parameters. Additional assumptions may be applied in models that inherit from this one.

Figure B.5 shows how instances of `Species` models are connected within a `Subregion`. A single species in a single phase is called a *configuration*. The generalized resistances (R) affect the force and rates of chemical exchange and heat flow associated with differences in activity, velocity, and temperature (respectively) between each configuration and a common node. These exchange processes are diffusive. Each resistor generates heat in the `Species` instance that contains it.

Figure B.6 shows how a configuration is connected between neighboring instances of a `Subregion`. Material, translational momentum, and thermal energy are transported by both advection and diffusion. Upstream discretization is applied if it is enabled via the `upstreamX`, etc. parameters. Like for exchange, the transport resistances are inside the `Species` model.

The `Species` instances within a `Phase` are combined by Dalton's law of partial pressures (see the Dalton connector), as shown in Figure B.7a. The pressures are additive, and each species is assumed to exist at the total extensive volume of the phase. Within a `Subregion`, the `Phases` are combined by Amagat's law of partial volumes (see the Amagat connector), as shown in Figure B.7b. The volumes are additive, and each species is assumed to exist at the total pressure in the subregion.

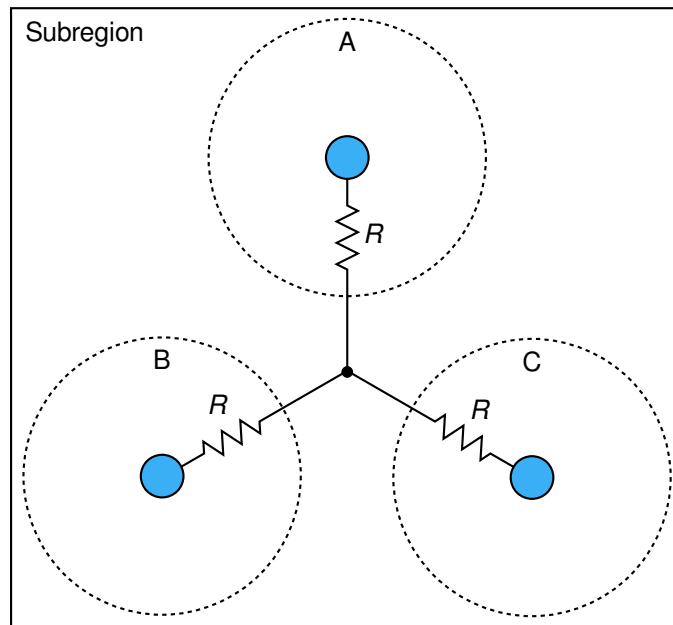


Figure B.5: Exchange of a quantity (material, translational momentum, or thermal energy) among configurations (A, B, and C) within a subregion.

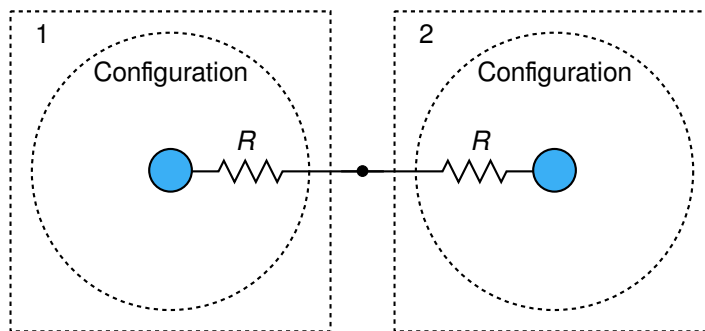
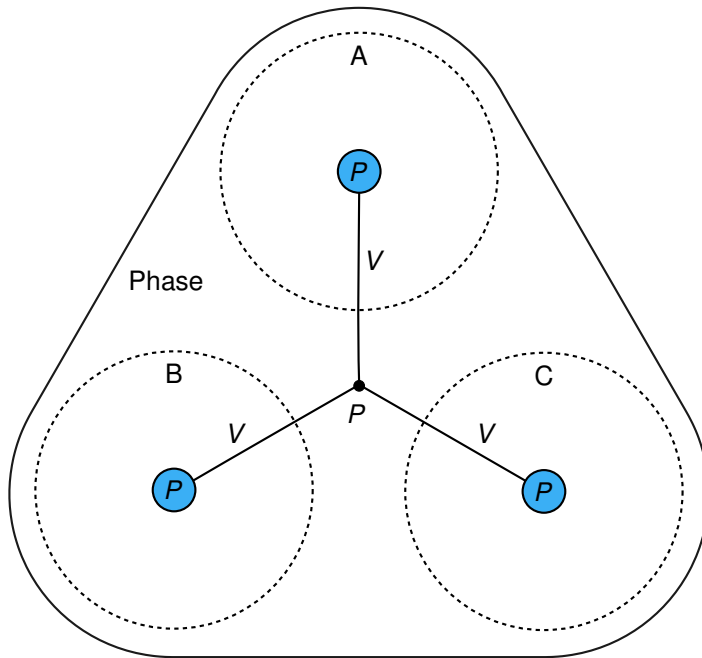
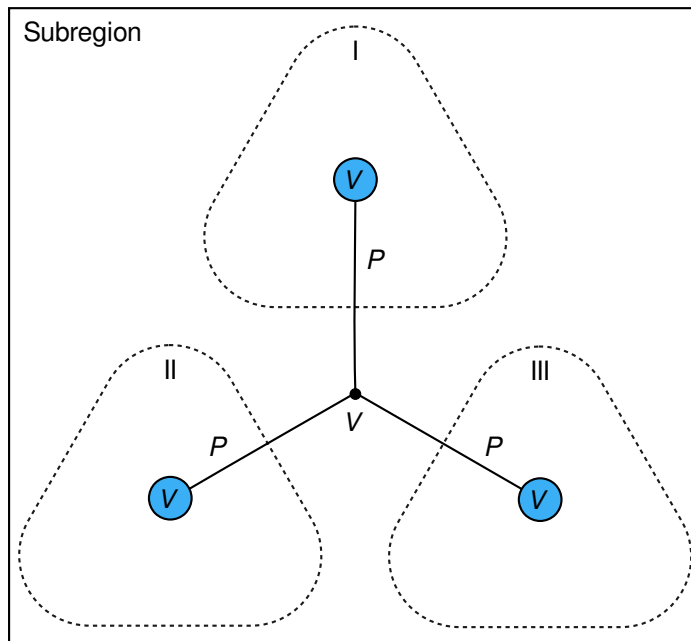


Figure B.6: Transport of a quantity associated with the same configuration between subregions (1 and 2).



(a) Pressures of species (A, B, and C) are additive within a phase.



(b) Volumes of phases (I, II, and III) are additive within a subregion.

Figure B.7: Methods of attributing pressure and volume.

Notes regarding the parameters:

1. The effect transport lengths (kL) may be different than the geometric lengths along the transport axes due to tortuosity. The tortuosity may be anisotropic.
2. If the interval for chemical exchange (τ'), mobility (μ), thermal conductivity (ν), fluidity (η), or thermal resistivity (θ) is zero, then it should usually be set as `final` so that index reduction may be performed. If two configurations are connected through their `intra`, `inter`, or `boundaries` connectors and both have zero generalized resistivities for a quantity, then index reduction [13] is necessary.
3. Even if an initialization parameter is not selected for explicit use, it may be used as a guess value.
4. If `ConsThermo.IC` is used for a state (via `consMaterial` or `consEnergy`), then the associated initial condition (IC) will be applied forever instead of the corresponding conservation equation.
5. If `consEnergy` is `ConsThermo.steady`, then $NT\partial s/\partial t + M\phi\partial\phi/\partial t$ is treated as zero and removed from the energy balance.
6. If a transport axis is not included (via the outer `inclTrans[:]` parameter which maps to `{inclTransX, inclTransY, inclTransZ}` in the Subregion model), then the associated pair of boundaries is removed from the model.
7. The `start` values of the initial conditions for pressure and temperature (p_{IC} and T_{IC}) are the global default pressure and temperature (via the `outer` instance of the `Environment` model). The `start` values of the initial conditions for other intensive properties (ρ_{IC} , h_{IC} , and g_{IC}) are related to the initial pressure and temperature by the characteristics of the species. The `start` value of the initial condition for the extensive volume (V_{IC}) is the volume of the subregion. The `start` value for particle number (N_{IC}) is related to it via the material characteristics and the initial pressure and temperature. In order to apply other values for any of these initial conditions, it may be necessary to do so before translating the model.
8. N_{min} establishes an amount of material below which no exchange occurs. It can help to avoid stiffness when very little of a species exists in a subregion.

In the `boundaries` connector array, the transverse translational flow ($m\dot{\Phi}$) is only the force due to diffusion. Translational advection is calculated from the velocity and the current. The thermal flow (Q) is only the rate of heat transfer due to diffusion. The advection of thermal energy is determined from the thermodynamic state at the boundary and the current.

For the variables that relate to transport, the first index is the axis and the second index is the side. The sides are ordered from negative to positive, according to the `Side` enumeration. Velocity and force are additionally indexed by the orientation of the momentum with respect to the boundary. The orientations are ordered following the normal axis in Cartesian space, according to the `Orient` enumeration.

Table B.64: Connectors of `FCSys.Species.Species`.

Type	Name	Description
Intra	<code>intra[n_intra]</code>	Connectors to exchange translational momentum and energy within the phase
Inter	<code>inter[n_inter]</code>	Connectors to exchange translational momentum and energy with all other species
Dalton	<code>dalton</code>	Connector for additivity of pressure
Material properties		
replaceable package Data	Characteristic data	

Table B.64 lists the connectors or connector variables of this class.

B.70.2 Modelica definition

```

partial model Species
  "Base model for one chemical species in one phase"

  import assert = FCSys.Utilities.assertEval;

  // Material properties
  replaceable package Data = Characteristics.BaseClasses.Characteristic
    constrainedby Characteristics.BaseClasses.Characteristic
    "Characteristic data";
  Q.Mobility mu(nominal=10*U.cm^2/(U.V*U.s)) = Data.mu(T, v) "Mobility";
  Q.TimeAbsolute nu(nominal=1e-9*U.s) = Data.nu(T, v) "Thermal independity";

  // Assumptions
  parameter Integer n_trans=1 "Number of transport axes";
  parameter Integer n_intra=0 "Number of exchange connections within the
    phase";

```

```

parameter Integer n_inter=0
    "Number of exchange connections with other phases";

// Initialization parameters
parameter Q.Amount N_IC(start=V_IC*rho_IC) "Initial amount of material";
parameter Q.Density rho_IC(start=1/Data.v_Tp(T_IC, p_IC)) "Initial
    density";
parameter Q.Volume V_IC(start=product(L)) "Initial volume";
parameter Q.PressureAbsolute p_IC(start=environment.p) "Initial pressure";
parameter Q.TemperatureAbsolute T_IC(start=environment.T)
    "Initial temperature";
parameter Q.Potential h_IC(start=Data.h(T_IC, p_IC), displayUnit="kJ/mol")
    "Initial specific enthalpy";
parameter Q.Potential g_IC(start=Data.g(T_IC, p_IC), displayUnit="kJ/mol")
    "Initial Gibbs potential";

// Independence factors
parameter Q.NumberAbsolute k_intra_Phi[n_intra, n_trans]=ones(n_intra,
    n_trans) "For translational exchange among species within the phase";
parameter Q.NumberAbsolute k_intra_Q[n_intra]=ones(n_intra)
    "For thermal exchange among species within the phase";

// Preferred states
// Note: The start values for these variable aren't fixed because the
// initial equation section will be used instead.
Q.Amount N(
    final min=Modelica.Constants.small,
    nominal=4*U.C,
    final start=N_IC,
    final fixed=false,
    stateSelect=StateSelect.prefer) "Amount of material";
Q.TemperatureAbsolute T(
    nominal=300*U.K,
    final start=T_IC,
    stateSelect=StateSelect.prefer) "Temperature";
Q.Velocity phi[n_trans](
    each nominal=100*U.cm/U.s,
    each start=0,
    each stateSelect=StateSelect.prefer) "Velocity";

// Aliases (for common terms)
// Note: StateSelect.never helps avoid dynamic state selection of these
// variables in Dymola 2014.
Q.PressureAbsolute p(
    nominal=U.atm,
    final start=p_IC,
    final fixed=false,
    stateSelect=StateSelect.never) "Pressure";
Q.Potential g(
    nominal=U.V,
    final start=g_IC,
    final fixed=false,
    stateSelect=StateSelect.never) "Specific Gibbs energy";
Q.Mass M(

```

```

nominal=1e-3*U.g,
final start=Data.m*N_IC,
stateSelect=StateSelect.never) "Mass";
Q.VolumeSpecific v(
nominal=U.cc/(4*U.C),
final start=1/rho_IC,
final fixed=false,
stateSelect=StateSelect.never) "Specific volume";
Q.Potential h(
nominal=U.V,
final start=h_IC,
final fixed=false,
stateSelect=StateSelect.never) "Specific enthalpy";
Q.NumberAbsolute s(
nominal=25,
final start=(h_IC - g_IC)/T_IC,
stateSelect=StateSelect.never) "Specific entropy";

// Auxiliary variables (for analysis)
// -----
// Thermodynamic properties
output Q.Density rho(stateSelect=StateSelect.never) = 1/v if
environment.analysis
"Density";
output Q.MassVolumic mrho(stateSelect=StateSelect.never) = Data.m*rho if
environment.analysis "Volumic mass";
output Q.Amount S(stateSelect=StateSelect.never) = N*s if
environment.analysis
"Entropy";
output Q.CapacityThermalSpecific c_p(stateSelect=StateSelect.never) =
Data.c_p(T, p) if environment.analysis "Isobaric specific heat capacity";
output Q.CapacityThermalSpecific c_v(stateSelect=StateSelect.never) =
Data.c_v(T, p) if environment.analysis "Isochoric specific heat
capacity";
output Q.PressureReciprocal beta(stateSelect=StateSelect.never) =
Data.beta(T,
p) if environment.analysis "Isothermal compressibility";
//
// Time constants
output Q.TimeAbsolute tau_PhiE_intra[n_intra, n_trans](
each stateSelect=StateSelect.never,
each start=U.s) = {Data.m*mu*k_intra_Phi[i, :] for i in 1:n_intra} if
environment.analysis and n_intra > 0
"Time constants for translational intra-phase exchange";
output Q.TimeAbsolute tau_PhiE_inter[n_inter, n_trans](
each stateSelect=StateSelect.never,
each start=U.s) = {Data.m*mu*k_inter_Phi[i, :] for i in 1:n_inter} if
environment.analysis and n_inter > 0
"Time constants for translational inter-phase exchange";
output Q.TimeAbsolute tau_QE_intra[n_intra](
each stateSelect=StateSelect.never,
each start=U.s) = c_p*nu*k_intra_Q if environment.analysis and n_intra >
0
"Time constants for thermal intra-phase exchange";

```

```

output Q.TimeAbsolute tau_QE_inter[n_inter](
    each stateSelect=StateSelect.never,
    each start=U.s) = c_p*nu*k_inter_Q if environment.analysis and n_inter >
    0
    "Time constants for thermal inter-phase exchange";
// Note: The structure of the problem should not change if these auxiliary
// variables are included (hence, StateSelect.never).
//
// Translational momentum balance
output Q.Force f_DE[n_trans](each stateSelect=StateSelect.never) =
    sum(intra[
        i].mPhidot for i in 1:n_intra) + sum(inter[i].mPhidot for i in 1:n_inter)
    if environment.analysis
    "Friction from other configurations (diffusive exchange)";
//
// Energy balance
output Q.Power Edot_DE(stateSelect=StateSelect.never) = sum(intra[i].phi*
    intra[i].mPhidot for i in 1:n_intra) + sum(inter[i].phi*inter[i].mPhidot
    for i in 1:n_inter) + sum(intra.Qdot) + sum(inter.Qdot) if
    environment.analysis
    "Rate of diffusion of energy from other configurations";
Connectors.Intra intra[n_intra](each final n_trans=n_trans, each T(final
    start=T_IC, final fixed=false))
    "Connectors to exchange translational momentum and energy within the
    phase";
Connectors.Inter inter[n_inter](each final n_trans=n_trans, each T(final
    start=T_IC, final fixed=false))
    "Connectors to exchange translational momentum and energy with all other
    species";

Connectors.Dalton dalton(V(
    min=0,
    final start=V_IC,
    final fixed=false), p(final start=p_IC, final fixed=false))
    "Connector for additivity of pressure";

// Geometric parameters
protected
outer Q.Volume V "Volume of the phase (not of the subregion)";
outer parameter Q.Length L[Axis] "Lengths of the subregion";
outer parameter Integer cartTrans[:];
    "Cartesian-axis indices of the transport axes";
outer parameter Q.NumberAbsolute k_inter_Phi[:, :];
    "Independence factors for translational exchange with other phases";
outer parameter Q.NumberAbsolute k_inter_Q[:];
    "Independence factors for thermal exchange among with other phases";

outer Conditions.Environment environment "Environmental conditions";

initial equation
    // Check the initial conditions.
    assert(V >= 0, "The volume of " + getInstanceName() + " is negative.
    Check that the volumes of the other phases are set properly.");

```

```

equation
  // Aliases (only to clarify and simplify other equations)
  h = g + T*s;
  p = dalton.p;
  v*N = dalton.V;
  M = Data.m*N;

  // Thermodynamic correlations
  if Data.isCompressible then
    p = Data.p_Tv(T, v);
  else
    v = Data.v_Tp(T, p);
  end if;
  h = Data.h(T, p);
  s = Data.s(T, p);

  // Exchange
  // -----
  // Within the phase
  for i in 1:n_intra loop
    k_intra_Phi[i, :]*mu .* intra[i].mPhidot = N*(intra[i].phi - phi)
      "Translational";
    k_intra_Q[i]*nu*intra[i].Qdot = N*(intra[i].T - T) "Thermal";
  end for;
  //
  // With other phases
  for i in 1:n_inter loop
    k_inter_Phi[i, :]*mu .* inter[i].mPhidot = N*(inter[i].phi - phi)
      "Translational";
    k_inter_Q[i]*nu*inter[i].Qdot = N*(inter[i].T - T) "Thermal";
  end for;
end Species;

```

B.71 FCSys.Subregions.Subregion



Subregion with all phases

B.71.1 Information

Assumptions:

1. The oxygen reduction reaction generates liquid water if it is included; otherwise, it generates H₂O vapor. Since phase change is a dynamic, nonequilibrium process, there is a difference.

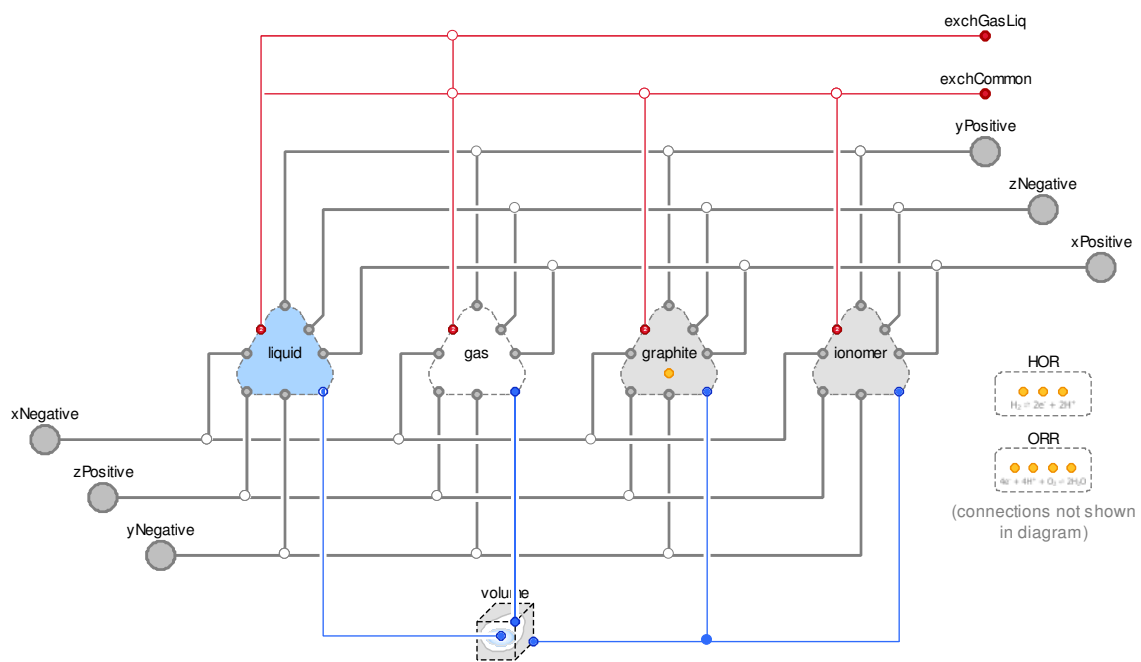


Figure B.8: Diagram of FCSys.Subregions.Subregion.

Figure B.8 shows the diagram of this class. Table B.65 lists its parameters. Table B.66 lists its connectors or connector variables.

This class extends from `PartialSubregion` (Base model for multi-dimensional, multi-species storage, transport, and exchange).

Table B.65: Parameters of `FCSys.Subregions.Subregion`.

Type	Name	Default	Description
CapillaryVolume	volume		Volume with capillary pressure included
		—Geometry—	
Length	L[Axis]	{U.cm,U.cm,U.cm}	Lengths [L]
		—Phases (click to edit)—	
Gas	gas		Gas
Graphite	graphite		Graphite
Ionomer	ionomer		Ionomer
Liquid	liquid		Liquid
		—Independence factors—	
ExchangeParams	common		Among all phases
ExchangeParams	gasLiq		Between gas and liquid
		—Assumptions—	
		—Included transport axes—	
Boolean	inclTransX	true	X
Boolean	inclTransY	true	Y
Boolean	inclTransZ	true	Z

Table B.66: Connectors of `FCSys.Subregions.Subregion`.

Type	Name	Description
BoundaryBus	xNegative	Negative boundary along the x axis
BoundaryBus	yNegative	Negative boundary along the y axis
BoundaryBus	zNegative	Negative boundary along the z axis

continued on the next page ...

... continued from the previous page

Type	Name	Description
BoundaryBus	xPositive	Positive boundary along the x axis
BoundaryBus	yPositive	Positive boundary along the y axis
BoundaryBus	zPositive	Positive boundary along the z axis

B.72 FCSys.Units

Constants and units of physical measure

B.72.1 Information

The `Units` package is abbreviated as `U` for convenience throughout the rest of `FCSys`. For example, an initial pressure might be defined as $p_{IC} = U \cdot \text{atm}$.

The information below has been updated from [186]. That paper also offers suggestions as to how the approach might be better integrated in Modelica. For more information, please also see the documentation of the `Quantities` package.

B.72.2 Overview

Models of physical systems involve variables that represent physical quantities. As stated by the Bureau International des Poids et Mesures (BIPM) [187, p. 103]:

“The value of a quantity is generally expressed as the product of a number and a unit. The unit is simply a particular example of the quantity concerned which is used as a reference, and the number is the ratio of the value of the quantity to the unit.”

In general, a unit may be the product of powers of other units, whether they are base units or units derived from the base units in the same manner.

In Modelica, a physical quantity is generally expressed as an instance of the `Real` type. Its `value` attribute is typically a number associated with the value of the quantity (not the value of the quantity directly). Its `unit` attribute is a string that describes the unit by which the value of the quantity has been divided to arrive at the number.¹ The `displayUnit` attribute (also a string) describes the unit by which the value of the quantity should be divided to arrive at the

number as it is entered by or presented to the user. The `Real` type contains other attributes as well, including quantity string.

The `SIunits` package of the Modelica Standard Library contains types that extend from the `Real` type. The type definitions modify the `unit`, `displayUnit`, and `quantity` attributes (among others) to represent various physical quantities. The `unit` and `displayUnit` attributes are based on the International System of Units (Système international d'unités, SI). The `quantity` string is generally used to describe the name of the physical quantity. For example, the `Velocity` type has a `unit` of "m/s" and a `quantity` of "Velocity". If an instance of `Velocity` has a value of one ($v = 1$), then it is meant that "the value of velocity is one meter per second." Again, the `value` attribute represents the number, or the value of the quantity divided by the unit, not the value of the quantity itself.

This apparent conflict is solved in `FCSys` by establishing units (including the meter and the second) as mathematical entities and writing $v = 1 \text{ m/s}$ (in code, $v = 1 * U.m / U.s$ or simply $v = U.m / U.s$). Here, the variable v directly represents the quantity. Its `value` attribute is truly the value of the quantity in the context of the statement by BIPM (above). One advantage is that unit conversion is handled naturally. The essence of unit conversion is the phrase "value of a quantity in a unit" typically means "value of a quantity divided by a unit." Continuing with the previous example, v is divided by m/s in order to display v in meters per second (as a number). If another unit of length like the foot is established by the appropriate relation ($1 \text{ ft} \approx 0.3048 \text{ m}$) and v is divided by ft/s, the result is velocity in feet per second (~ 3.2894). Some units such as $^{\circ}\text{C}$, `Pag`, and `dB` involve offsets or nonlinear transformations between the value of the quantity and the number; these are described by functions besides simple division.

As another example, frequency is sometimes represented by a variable in hertz or cycles per second (e.g., ν) and other times by a variable in radians per second (e.g., ω). If the variable represents the quantity directly, then there is no need to specify which units it is in. The units are included; they have not been factored out by division (or another function). A common variable (e.g., f) can be used in both cases, which simplifies and standardizes the equations of a model. The forms are algebraically equivalent due to the relationships among units (e.g., $1 \text{ cycle} = 2\pi \text{ rad}$).

B.72.3 Method

In FCSys, each unit is a constant quantity. The values of the units, like other quantities, is the product of a number and a unit. Therefore, units may be derived from other units (e.g., cycle = 2π rad). This recursive definition leaves several units (in SI, 7) that are locally independent and must be established universally. These base units are established by the "particular example of the quantity concerned which is used as a reference" quoted previously [187]. The choice of the base units is somewhat arbitrary [34, p. 375], but regardless, there are a number of units that must be defined by example.

If only SI will be used, then it is easiest to set each of the base units of SI equal to one—the meter (m), kilogram (kg), second (s), ampere (A), kelvin (K), mole (mol), and candela (cd). This is implicitly the case in the `SIunits` package, but again, it hardly captures the idea that the value of a quantity is the product of a number and a unit.

Instead, in FCSys, most of the base units are established by universal physical constants. The "particular example of the quantity" [187] is an experiment that yields precise and universally repeatable results in determining a constant rather than a prototype (e.g., the International Prototype of the Kilogram) which is carefully controlled and distributed via replicas. This method of defining the base units from measured physical quantities (rather than vice versa) is natural and reflects the way that standards organizations (e.g., NIST) define units. A system of units is considered to be natural if all of its base units are established by universal physical constants. Often, those universal constants are defined to be equal to one, but the values can be chosen to scale the numerical values of variables during simulation.

There are physical systems where typical quantities are many orders of magnitude larger or smaller than the related product of powers of base SI units (e.g., the domains of astrophysics and atomic physics). In modeling and simulating those systems, it may be advantageous to choose appropriately small or large values (respectively) for the corresponding base units such that the product of the number (large or small in magnitude) and the unit (small or large, respectively) is well-scaled. Products of this type are often involved in initial conditions or parameter expressions, which are not time-varying. Therefore, the number and the unit can be

multiplied before the dynamic simulation. During the simulation, only the value is important. After the simulation, the trajectory of the value may be divided by the unit for display. This scaling is usually unnecessary due to the wide range and appropriate distribution of the real numbers that are representable in floating point space. The Modelica language specification recommends that floating point numbers be represented in at least IEEE double precision, which covers magnitudes from $\sim 2.225 \times 10^{-308}$ to $\sim 1.798 \times 10^{308}$ [193, p. 13]. However, in some cases it may be preferable to carefully scale the units and use single precision instead for the sake of computational performance. There are fields of research where, even today, simulations are sometimes performed in single precision [231, 232] and where scaling is a concern [233, p. 29].

The method is neutral with regards to not only the values of the base units, but also the choice of the base units and even the number of base units. This is an advantage because many systems of units are used in science and engineering besides SI. As mentioned previously, the choice of base units is somewhat arbitrary, and different systems of units are based on different choices. Some systems of units have fewer base units (lower rank) than SI, since additional constraints are added that exchange base units for derived units. For example, the Planck, Stoney, Hartree, and Rydberg systems of units define the Boltzmann constant to be equal to one ($k_B = 1$) [http://en.wikipedia.org/wiki/Natural_units]. The unit K is eliminated [234, p. 386] or, more precisely, considered a derived unit instead of a base unit. In SI, the kelvin would be derived from the units kilogram, meter, and second ($K \approx 1.381 \times 10^{-23} \text{ kg m}^2/\text{s}^2$).

There are six independent constants or units in the `Units` package (see `Units.Bases`), but SI has seven independent base units (m, kg, s, A, K, mol, and cd). In `FCSys`, two additional constraints are imposed in order to simplify the model equations and allow electrons and chemical species to be represented by the same base `Species` model. First, the Faraday constant (k_F or 96485.3399 C/mol) is normalized to one. This implies that the mole (mol) is proportional to the coulomb (C), which is considered a number of reference particles given the charge number. Also, the gas constant (R or 8.314472 J/(mol K)) is normalized to one. Therefore, the kelvin (K) is proportional to the volt (V or J/C). In addition, the radian (rad) is

defined as a base constant. However, it must be set equal to one in the current version of the International System of Units (SI) [187].

B.72.4 Implementation

The units and constants are defined as variables in this `Units` package. Each is a `constant` of the appropriate type from the `Quantities` package. The first section of the Modelica definition of this package establishes mathematical constants. The next section establishes the base constants and units, which grouped in a replaceable subpackage. The third section establishes the constants and units which may be derived from the base units and constants using accepted empirical relations. The rest of the code establishes the SI prefixes and the remaining derived units and constants. The SI prefixes are included in their unabbreviated form in order to avoid name conflicts. All of the primary units of SI are included (Tables 1 and 3 of [187]) except for °C, since it involves an offset. Other convenient units are included for the system at hand (e.g., atm).

The `Units.setup` function establishes unit conversions using the values of the units, constants, and prefixes. These unit conversions may include offsets. The function also sets the default display units. It is automatically called when `FCSys` is loaded from the "FCSys/load.mos" script. It can also be called manually from the "Re-initialize the units" command available in Dymola from the `Units` package or any subpackage. A spreadsheet (Resources/quantities.xls) is available to help maintain the quantities, default units, and the setup function.

The values of the units, constants, and prefixes can be evaluated by translating the `Units-Examples.Evaluate` model. This defines the values in the Dymola workspace. For convenience, the "FCSys/load.mos" script automatically does this and saves the result as "units.mos" in the working directory.

This package also contains functions (e.g., `to_degC`) that convert quantities from the unit system defined in `FCSys` to quantities expressed in units. These functions are included for units that involve offsets. For conversions that require just a scaling factor, it is simpler to use the units directly. For example, to convert from potential in volts use $v = v_V * U.V$, where v is potential and v_V is potential expressed in volts.

An instance of the `Environment` model is usually included at the top level of a model. It records the base units or constants so that it is possible to re-derive all of the other units and constants. This is important in order to properly interpret simulation results if the base units or constants are later re-adjusted.








Where the `der` operator is used in models, it is explicitly divided by the unit second (e.g., `der(x)/U.s`). This is necessary because the global variable `time` is in seconds (i.e., `time` is a number, not a quantity).

For convenience, some units that include prefixes are defined (e.g., kg, mm, and kPa). However, most prefixes must be given as explicit factors (e.g., `U.kilo*U.m`).

Although it is not necessary due to the acausal nature of Modelica, the declarations in this package are sorted so that they can be easily ported to imperative or causal languages (e.g., Python and C). In fact, this has been implemented in the included `FCRes` module for plotting and analysis.

Table B.67 lists the contents of this class.

Table B.67: Contents of the `FCSys.Units` package.

Name	Description
 <code>setup</code>	Establish conversions to display quantities in units
 <code>Examples</code>	Examples
 <code>Bases</code>	Sets of base constants and units
 <code>from_degC</code>	Convert from temperature in degree Celsius to temperature as a quantity
 <code>to_degC</code>	Convert from temperature as a quantity to temperature in degree Celsius
 <code>from_kPag</code>	Convert from gauge pressure in kilopascals to absolute pressure as a quantity
 <code>to_kPag</code>	Convert from absolute pressure as a quantity to gauge pressure in kilopascals
<code>pi=2*acos(0)</code>	pi (π)
<code>base</code>	Scalable base constants and units
<code>rad=base.rad</code>	radian

continued on the next page . . .

... continued from the previous page

Name	Description
$R_{\infty} = \text{base}.R_{\infty}$	Rydberg constant (R_{∞})
$c = \text{base}.c$	speed of light in vacuum (c)
$k_J = \text{base}.k_J$	Josephson constant (k_J)
$R_K = \text{base}.R_K$	von Klitzing constant (R_K)
'cd' = base.'cd'	candela
$k_F = \text{base}.k_F$	Faraday constant (k_F)
$R = \text{base}.R$	gas constant
$m = 10973731.568539 \cdot \text{rad}/R_{\infty}$	meter
$s = 299792458 \cdot \text{m}/c$	second
$\text{Wb} = 483597.870\text{e}9/k_J$	weber
$S = 25812.8074434/R_K$	siemen
$\text{mol} = 96485.3365 \cdot \text{Wb} \cdot \text{S}/k_F$	mole
$K = 8.3144621 \cdot (\text{Wb} \cdot \text{rad})^2 \cdot \text{S}/(\text{s} \cdot \text{mol} \cdot R)$	kelvin
$V = \text{Wb} \cdot \text{rad}/\text{s}$	volt
$A = V \cdot S$	ampere
$C = A \cdot \text{s}$	coulomb
$J = V \cdot C$	joule
$\text{Gy} = (\text{m}/\text{s})^2$	gray
$\text{kg} = \text{J}/\text{Sv}$	kilogram
yotta=1e24	yotta (Y)
zetta=1e21	zetta (Z)
exa=1e18	exa (E)
peta=1e15	peta (P)
tera=1e12	tera (T)
giga=1e9	giga (G)
mega=1e6	mega (M)
kilo=1e3	kilo (k)
hecto=1e2	hecto (h)

continued on the next page ...

... continued from the previous page

Name	Description
deca=1e1	deca (da)
deci=1e-1	deci (d)
centi=1e-2	centi (c)
milli=1e-3	milli (m)
micro=1e-6	micro (u)
nano=1e-9	nano (n)
pico=1e-12	pico (p)
femto=1e-15	femto (f)
atto=1e-18	atto (a)
zepto=1e-21	zepto (z)
yocto=1e-24	yocto (y)
cyc=2*pi*rad	cycle
Hz=cyc/s	hertz
sr=rad ²	steradian
N=J/m	newton
Pa=N/m ²	pascal
W=J/s	watt
F=C/V	farad
ohm=1/S	ohm (Ω)
H=V*s/A	henry
T=Wb/m ²	tesla
lm='cd'*sr	lumen
lx=lm/m ²	lux
Bq=Hz	becquerel
Sv=Gy	sievert
kat=mol/s	katal
g=kg/kilo	gram
min=60*s	minute

continued on the next page ...

... continued from the previous page

Name	Description
$hr=60*min$	hour
$day=24*hr$	day
$degree=2*pi*rad/360$	degree (°)
$L=(deci*m)^3$	liter (L or l)
$G_0=2/R_K$	conductance quantum (G_0)
$\Phi_0=1/k_J$	magnetic flux quantum (Φ_0)
$q=G_0*\Phi_0$	elementary charge
$h=2*q*\Phi_0$	Planck constant
$alpha=pi*1e-7*c*s*G_0/(m*S)$	fine-structure constant (α)
$Z_0=2*R_K*alpha$	characteristic impedance of vacuum (Z_0)
$mu_0=Z_0/c$	magnetic constant (μ_0)
$epsilon_0=1/(Z_0*c)$	electric constant (ϵ_0)
$k_A=mu_0/(4*pi)$	magnetic force constant (k_A)
$k_e=k_A*c^2$	Coulomb constant (k_e)
$E_h=2*R_{inf}*h*c$	Hartree energy (E_h)
$eV=q*V$	electron volt
$N_A=k_F/q$	Avogadro constant (N_A)
$k_B=R/N_A$	Boltzmann constant (k_B)
$c_1=cyc*h*c^2$	first radiation constant (c_1)
$c_2=h*c/k_B$	second radiation constant (c_2)
$c_3_{lambda}=c_2/4.965114231744276$	Wien wavelength displacement law constant ($c_{3\lambda}$)
$c_3_f=2.821439372122079*k_B/h$	Wien frequency displacement law constant (c_{3f})
$sigma=2*pi*(k_B*pi)^4/(15*(h*rad)^3*c^2)$	Stefan-Boltzmann constant (σ)
$bar=1e5*Pa$	bar
$angstrom=0.1*nano*m$	angstrom (Å)
$atm=101325*Pa$	atmosphere
$kPa=kilo*Pa$	kilopascal

continued on the next page ...

... continued from the previous page

Name	Description
<code>kJ=kilo*J</code>	kilojoule
<code>cm=centi*m</code>	centimeter
<code>mm=milli*m</code>	millimeter
<code>mA=milli*A</code>	milliampere
<code>um=micro*m</code>	micrometer
<code>ms=milli*s</code>	millisecond
<code>'%'=centi</code>	percent (%)
<code>M=U.mol/U.L</code>	molar
<code>cc=U.cm³</code>	cubic centimeter

B.73 FCSys.Units.Bases

Sets of base constants and units

B.73.1 Information

FCSys requires that the Faraday and gas constants are normalized to one. The structure of the `Units` package allows those constants to be relaxed, but the models in FCSys generally do not.















Some natural systems of units are not compatible with FCSys. Since the Faraday and gas constants are both normalized, it follows that $k_B = q$. This is not the case for the Planck, Rydberg, and Natural systems of units [http://en.wikipedia.org/wiki/Natural_units].

The quasi-SI sets in this package are named by listing (in alphabetical order) the two units that are *not* normalized for the sake of setting the Faraday and gas constants equal to one. There are eight possible sets of this type (SIAK, SIAM, SIAs, SIKmol, SIKs, SImmol SImS, SImols).

For more information, please see the documentation for the `Units` package.

Table B.68 lists the contents of this class.

Table B.68: Contents of the `FCSys.Units.Bases` package.

Name	Description
 FC	Base constants and units that are well-scaled for fuel cell simulation and analysis
 Gaussian	Base constants and units for Gaussian units ($k_A = k_e = 1$)
 Hartree	Base constants and units for Hartree units
 LH	Base constants and units for Lorentz-Heaviside units ($\mu_0 = \epsilon_0 = 1$)
 Stoney	Base constants and units for Stoney units
 SIAK	Base constants and units for SI with k_F and R normalized instead of A and K
 SIAm	Base constants and units for SI with k_F and R normalized instead of A and m
 SIAs	Base constants and units for SI with k_F and R normalized instead of A and s
 SIKmol	Base constants and units for SI with k_F and R normalized instead of K and mol
 SIKs	Base constants and units for SI with k_F and R normalized instead of K and s
 SImmol	Base constants and units for SI with k_F and R normalized instead of m and mol
 SImms	Base constants and units for SI with k_F and R normalized instead of m and s
 SImols	Base constants and units for SI with k_F and R normalized instead of mol and s
 Base	Base constants and units

B.74 `FCSys.Units.Bases.Base`



Base constants and units

B.74.1 Information

Please see the notes in the Modelica code and the documentation of the `Units` package.

B.74.2 Modelica definition

```

record Base "Base constants and units"
  extends Modelica.Icons.Record;

  final constant Q.Angle rad=1 "radian";
  // SI unit of rotation or planar angle
  // This condition is required by BIPM [BIPM2006, Table 3]. It can't be
  // relaxed because BIPM doesn't explicitly use angle in the definitions of
  // Hz, sr, etc. and NIST doesn't explicitly use angle in the relations for
  // R_inf, c_3_nu, etc. [NIST2010].
  constant Q.Wavenumber R_inf=1
    "Rydberg constant";
  // The SI unit length (meter) is inversely proportional to this value,
  // which should be increased for larger characteristic lengths.
  constant Q.Velocity c=1 "speed of light in vacuum (c)";
  // The SI unit time (second) is inversely proportional to this value (and
  // R_inf), which should be increased for larger characteristic times.
  constant Q.MagneticFluxReciprocal k_J=1
    "Josephson constant";
  // The SI unit of magnetic flux (weber) is inversely proportional to this
  // value, which should be increased for larger magnetic flux numbers.
  // Also, the SI unit of charge (coulomb) is inversely proportional to this
  // value.
  constant Q.ResistanceElectrical R_K=1
    "von Klitzing constant";
  // The SI unit of electrical conductance (siemen) is inversely proportional
  // to this value, which should be increased for larger characteristic
  // conductances. Also, the SI unit of charge (coulomb) is inversely
  // proportional to this value.
  constant Q.PowerRadiant 'cd'=1 "candela";
  // SI unit of luminous intensity
  // From http://en.wikipedia.org/wiki/Luminous\_intensity, accessed 11/5/11:
  // "luminous intensity is a measure of the wavelength-weighted power
  // emitted by a light source in a particular direction per unit solid
  // angle, based on the luminosity function, a standardized model of the
  // sensitivity of the human eye."
  final constant Q.Number k_F=1 "Faraday constant";
  // The unit of substance (mole) is inversely proportional to this value.
  // The Faraday constant isn't adjustable because the equations of FCSys
  // require that it's one, which means that charge is considered to be an
  // amount of substance.
  final constant Q.Number R=1 "gas constant";
  // The unit of temperature (kelvin) is inversely proportional to this
  // value. The gas constant isn't adjustable because the equations of FCSys
  // require that it's one, which means that temperature is considered to be
  // a potential.

end Base;

```

B.75 FCSys.Utilities

General supporting functions

B.75.1 Information

Table B.69 lists the contents of this class.

Table B.69: Contents of the `FCSys.Utilities` package.

Name	Description
<input type="checkbox"/> Chemistry	Functions to support chemistry
<input type="checkbox"/> Coordinates	Functions to handle Cartesian coordinates
<input type="checkbox"/> Means	Package of mathematical mean functions
<input type="checkbox"/> Polynomial	Polynomial functions
<input type="checkbox"/> Time	Functions to check translation time
<input type="radio"/> <code>arrayBooleanEqual</code>	Check if two <code>Boolean</code> vectors are equal
<input type="radio"/> <code>arrayIntegerEqual</code>	Check if two <code>Integer</code> vectors are equal
<input type="radio"/> <code>arrayRealEqual</code>	Check if two <code>Real</code> vectors are nearly equal
<input type="radio"/> <code>arrayStringEqual</code>	Check if two vectors of strings are equal
<input type="radio"/> <code>assertEval</code>	Assert function that forces Dymola to parse the message
<input type="radio"/> <code>Delta</code>	Return the second entry of a vector minus the first (Δ)
<input type="radio"/> <code>inSign</code>	Return the mathematical sign for the direction into a side or boundary
<input type="radio"/> <code>mod1</code>	Modulo operator for 1-based indexing (compatible with Mod- elica)
<input type="radio"/> <code>plot6</code>	Create six plots
<input type="radio"/> <code>round</code>	Round a <code>Real</code> variable to the nearest integer
<input type="radio"/> <code>selectBooleans</code>	Reduce a <code>Boolean</code> vector by selecting indices
<input type="radio"/> <code>selectIntegers</code>	Reduce an <code>Integer</code> vector by selecting indices
<input type="radio"/> <code>Sigma</code>	Return the sum of the first and second entries of a vector (Σ)




B.76 `FCSys.Utilities.Coordinates`

Functions to handle Cartesian coordinates

B.76.1 Information

Table B.70 lists the contents of this class.

Table B.70: Contents of the `FCSys.Utilities.Coordinates` package.

Name	Description
 <code>after</code>	Return the axis following a given axis in Cartesian coordinates
 <code>before</code>	Return the axis preceding a given axis in Cartesian coordinates
 <code>cartWrap</code>	Return the index to a Cartesian axis (1 to 3 or <code>Axis.x</code> to <code>Axis.z</code>) after wrapping






B.77 `FCSys.Utilities.Polynomial`

Polynomial functions

B.77.1 Information

Table B.71 lists the contents of this class.

Table B.71: Contents of the `FCSys.Utilities.Polynomial` package.

Name	Description
 <code>F</code>	$\int f() \cdot dx$ evaluated at x with zero integration constant
 <code>dF</code>	Derivative of <code>F()</code>
 <code>f</code>	Polynomial expressed in form: $f = ((\dots + a_{1-n})/x + a_n)/x + a_{1-n} + x \cdot (a_{2-n} + x \cdot (a_{3-n} + \dots))$
 <code>df</code>	Derivative of <code>f()</code>
 <code>d2f</code>	Derivative of <code>df()</code>

REFERENCES

- [1] MODELICA ASSOCIATION, “Modelica: A unified object-oriented language for physical systems modeling: Language specification,” May 2012. Version 3.3.
- [2] MODELICA ASSOCIATION, “Modelica Standard Library.” <http://www.modelica.org/libraries/Modelica>, Oct. 2010. Version 3.2.
- [3] OLIPHANT, T. E., “Python for scientific computing,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 10–20, 2007.
- [4] HUNTER, J. D., “Matplotlib: A 2D Graphics Environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [5] THOMAS, R. E. and ROSA, A. J., *The Analysis and Design of Linear Circuits*. Prentice Hall, 2nd ed., 1998.
- [6] MATEI, I. and BOCK, C., “Modeling methodologies and simulation for dynamical systems,” NIST Interagency/Internal Report (NISTIR) 7875, National Institute of Standards and Technology (NIST), Gaithersburg, MD, Aug. 2012.
- [7] WESTERBERG, A. W. and PIELA, P. C., “Equation-based process modeling,” tech. rep., Engineering Design Research Center (EDRC), Carnegie Mellon University, 1994.
- [8] ZENITH, F., SELAND, F., KONGSTEIN, O. E., BORRESEN, B., TUNOLD, R., and SKOGESTAD, S., “Control-oriented modelling and experimental study of the transient response of a high-temperature polymer fuel cell,” *Journal of Power Sources*, vol. 162, no. 1, pp. 215–227, 2006.
- [9] WILLEMS, J. C., “The behavioral approach to open and interconnected systems: Modeling by zooming, tearing, and linking,” *IEEE Control Systems Magazine*, pp. 46–99, Dec. 2007.
- [10] CELLIER, F. E., ELMQVIST, H., and OTTER, M., “Modeling from physical principles,” in *The Control Handbook*, pp. 99–107, CRC Press, 1996.
- [11] KOFRÁNEK, J., MATEJÁK, M., PRIVITZER, P., and TRIBULA, M., “Causal or acausal modelling: Labour for humans or labour for machines,” in *Technical Computing Conference Proceedings*, (Prague), 2008.
- [12] CELLIER, F. E. and KOFMAN, E., *Continuous System Simulation*. Springer, 2006.
- [13] MATTSSON, S. E. and SODERLIND, G., “Index reduction in differential-algebraic equations using dummy derivatives,” *SIAM Journal on Scientific Computing*, vol. 14, pp. 677–692, May 1993.
- [14] DONIDA, F., CASELLA, F., and FERRETTI, G., “Model order reduction for object-oriented models: A control systems perspective,” *Mathematical and Computer Modelling of Dynamical Systems*, vol. 16, pp. 269–284, Jun. 2010.

- [15] MODELICA ASSOCIATION, *Modelica: A Unified Object-Oriented Language for Physical Systems Modeling: Tutorial*. Linköping, Sweden, Dec. 2000. Version 1.4.
- [16] CELLIER, F. E. and GREIFENEDER, J., “Modeling chemical reactions in Modelica by use of chemo-bonds,” in *Proceedings of the 7th International Modelica Conference* (CASELLA, F., ed.), (Como, Italy), Modelica Association, Linköping University Electronic Press, Sep. 2009.
- [17] LARMINIE, J. and DICKS, A., *Fuel Cell Systems Explained*. John Wiley & Sons, 2nd ed., 2003.
- [18] BURKE, K. A., “Unitized regenerative fuel cell system development,” NASA report TM—2003-212739, Glenn Research Center, Cleveland, OH, Dec. 2003.
- [19] BRADLEY, T. H., *Model, Design and Energy Management of Fuel Cell Systems for Aircraft*. PhD thesis, Georgia Institute of Technology, Dec. 2008.
- [20] FAGHRI, A. and GUO, Z., “Challenges and opportunities of thermal management issues related to fuel cell technology and modeling,” *International Journal of Heat and Mass Transfer*, vol. 48, no. 19–20, pp. 3891–3920, 2005.
- [21] U.S. DEPARTMENT OF ENERGY, “Hydrogen, fuel cells & infrastructure technologies program: Multi-year research, development and demonstration plan,” tech. rep., Energy Efficiency and Renewable Energy, Oct. 2007. Section 3.4: Fuel Cells.
- [22] DING, X., DIDARI, S., FULLER, T. F., and HARRIS, T. A. L., “A new fabrication technique to manufacture an MEA using direct coating of Nafion[®] onto catalyzed GDL,” *Electrochemical Society Transactions*, vol. 33, no. 1 part 1, pp. 255–265, 2010.
- [23] CELLIER, F. E., *Continuous System Modeling*. Springer-Verlag, 1991.
- [24] MATZOPOULOS, M., “Advanced modelling accelerates fuel cell development,” *Fuel Cell Focus*, pp. 44–47, Sep. 2007.
- [25] PYSTER, A., OLWELL, D., HUTCHISON, N., ENCK, S., ANTHONY, J., HENRY, D., and SQUIRES, A., eds., *Guide to the Systems Engineering Body of Knowledge (SEBoK)*, vol. 2, ch. Representing Systems with Models, pp. 77–98. Hoboken, NJ: The Trustees of the Stevens Institute of Technology, Sep. 2012. Version 1.0.1.
- [26] SCHMITTINGER, W. and VAHIDI, A., “A review of the main parameters influencing long-term performance and durability of PEM fuel cells,” *Journal of Power Sources*, vol. 180, pp. 1–14, May 2008.
- [27] HALLINAN, D. T., DE ANGELIS, M. G., GIACINTI BASCHETTI, M., SARTI, G. C., and ELABD, Y. A., “Non-Fickian diffusion of water in Nafion,” *Macromolecules*, vol. 43, no. 10, pp. 4667–4678, 2010.
- [28] WAGNER, N., SCHNURNBERGER, W., MUELLER, B., and LANG, M., “Electrochemical impedance spectra of solid-oxide fuel cells and polymer membrane fuel cells,” *Electrochimica Acta*, vol. 43, no. 24, pp. 3785–3793, 1998.
- [29] MATTIUSI, C., “The finite volume, finite element, and finite difference methods as numerical methods for physical field problems,” vol. 113 of *Advances in Imaging and Electron Physics*, pp. 1–146, Elsevier Academic Press, 2000.

- [30] ZIMMER, D., *Equation-Based Modeling of Variable-Structure Systems*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zürich, 2010.
- [31] BROMAN, D., *Meta-Languages and Semantics for Equation-Based Modeling and Simulation*. PhD thesis, Linköping University, Linköping, Sweden, Oct. 2010.
- [32] ELMQVIST, H. and MATTSSON, S. E., “Modelica: The next generation modeling language an international design effort,” in *Proceedings of the 1st World Congress on System Simulation (WCSS’97)*, (Singapore), Sep. 1997.
- [33] ÅSTRÖM, K. J., ELMQVIST, H., and MATTSSON, S. E., “Evolution of continuous-time modeling and simulation,” in *Proceedings of the 12th European Simulation Multiconference (ESM’98)*, (Manchester, UK), pp. 9–18, 1998.
- [34] FRITZSON, P., *Principles of Object-Oriented Modeling and Simulation with Modelica 2.1*. Piscataway, NJ: IEEE Press, 2004.
- [35] SALDAMLI, L., *PDEModelica: A High-Level Language for Modeling with Partial Differential Equations*. PhD thesis, Linköping University, Jun. 2006.
- [36] DSHABAROW, F., “Support for Dymola in the modeling and simulation of physical systems with distributed parameters,” Master’s thesis, Swiss Federal Institute of Technology (ETH), Zürich, Jun. 2007.
- [37] DSHABAROW, F., CELLIER, F. E., and ZIMMER, D., “Support for Dymola in the modeling and simulation of physical systems with distributed parameters,” in *Proceedings of the 6th International Modelica Conference*, (Bielefeld, Germany), pp. 683–690, Modelica Association, Mar. 2008.
- [38] CHRISTEN, E. and BAKALAR, K., “VHDL-AMS: A hardware description language for analog and mixed-signal applications,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 46, no. 10, pp. 1263–1272, 1999.
- [39] IEEE COMPUTER SOCIETY, “IEEE standard VHDL analog and mixed-signal extensions,” 2007. IEEE Std 1076.1-2007.
- [40] ACCELLERA ORGANIZATION, INC., “Verilog-AMS language reference manual,” Jun. 2009. Version 2.3.1.
- [41] PÊCHEUX, F., LALLEMENT, C., and VACHOUX, A., “VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, pp. 204–225, Feb. 2005.
- [42] BARTON, P. I., *Modeling and Simulation of Combined Discrete/Continuous Processes*. PhD thesis, Imperial College of Science, Technology and Medicine, London, May 1992.
- [43] OH, M. and PANTELIDES, C., “Modelling and simulation language for combined lumped and distributed parameter systems,” *Computers and Chemical Engineering*, vol. 20, no. 6-7, pp. 611–633, 1996.
- [44] THE MATHWORKS INC., “Simscape,” Aug. 2012. Product brochure; available at <http://www.mathworks.com/products/simscape>.

- [45] THE MATHWORKS INC., *SimPowerSystems 5 User's Guide*. Natick, MA, Sep. 2010.
- [46] THE MATHWORKS INC., "The MathWorks introduces Simscape language for physical modeling." <http://www.mathworks.com/company/newsroom/The-MathWorks-Introduces-Simscape-Language-for-Physical-Modeling.html>, Oct. 2008. Accessed Aug. 23, 2013.
- [47] NILSSON, E. L. and FRITZSON, P., "BioChem: A biological and chemical library for Modelica," in *Proceedings of the 3rd International Modelica Conference*, (Linköping, Sweden), pp. 215–220, Modelica Association, Nov. 2003.
- [48] NILSSON, E. L. and FRITZSON, P., "Biochemical and metabolic modeling and simulation with Modelica," in *Proceedings of BioMedSim'2005*, (Linköping, Sweden), May 2005.
- [49] ELSHEIKH, A., "ADGenKinetics: An algorithmically differentiated library for biochemical networks modeling via simplified kinetics formats," in *Proceedings of the 9th International Modelica Conference*, (Munich, Germany), pp. 915–926, Modelica Association, Sep. 2012.
- [50] PAREDIS, C. J., "An open-source Modelica library of fluid power models," in *Bath/ASME Symposium on Fluid Power and Motion Control*, 2008.
- [51] PATANKAR, S. V., *Numerical Heat Transfer and Fluid Flow*. Computational Methods in Mechanics and Thermal Science, Levittown, PA: Taylor & Francis, 1980.
- [52] ELMQVIST, H., TUMMESCHEIT, H., and OTTER, M., "Object-oriented modeling of thermo-fluid systems," in *Proceedings of the 3rd International Modelica Conference*, (Linköping, Sweden), pp. 269–286, Modelica Association, Nov. 2003.
- [53] FRANKE, R., CASELLA, F., OTTER, M., SIELEMANN, M., ELMQVIST, H., MATTSON, S. E., and OLSSON, H., "Stream connectors: An extension of Modelica for device-oriented modeling of convective transport phenomena," in *Proceedings of the 7th International Modelica Conference*, (Como, Italy), pp. 108–121, Modelica Association, Sep. 2009.
- [54] HEFNI, B. E., BOUSKELA, D., and LEBRETON, G., "Dynamic modelling of a combined cycle power plant with ThermoSysPro," in *Proceedings of the 8th International Modelica Conference*, (Dresden, Germany), Modelica Association, Mar. 2011.
- [55] CASELLA, F. and LEVA, A., "Modelling of thermo-hydraulic power generation processes using Modelica," *Mathematical and Computer Modeling of Dynamical Systems*, vol. 12, pp. 19–33, Feb. 2006.
- [56] OLSSON, H., OTTER, M., MATTSSON, S. E., and ELMQVIST, H., "Balanced models in Modelica 3.0 for increased model quality," in *Proceedings of the 6th International Modelica Conference*, (Bielefeld, Germany), pp. 21–33, Modelica Association, Mar. 2008.
- [57] BROENINK, J. F., "Object-oriented modeling with bond graphs and Modelica," in *Proceedings of the International Conference on Bond Graph Modeling and Simulation (ICBGM)*, (San Francisco, CA), Jan. 1999.
- [58] CELLIER, F. E. and GREIFENEDER, J., "ThermoBondLib: A new Modelica library for modeling convective flows," in *Proceedings of the 6th International Modelica Conference*, (Bielefeld, Germany), pp. 163–172, Modelica Association, Mar. 2008.

- [59] GREIFENEDER, J. and CELLIER, F. E., “Modeling chemical reactions using bond graphs,” in *Proceedings of the 10th International Conference on Bond Graph Modeling and Simulation, ICBGM’12, Part of SummerSim 2012 Multiconference*, vol. 44, (Genoa, Italy), pp. 110–121, 2012.
- [60] HOGAN, N., “Heat transfer and the second law.” Course notes for *Modeling and Simulation of Dynamic Systems* (2.141), Massachusetts of Technology, Oct. 2006.
- [61] BRUUN, K., *Bond Graph Modelling of Fuel Cells for Marine Power Plants*. PhD thesis, Norwegian University of Science and Technology, Sep. 2009.
- [62] ZUPANCIC, B. and SODJA, A., “Computer-aided physical multi-domain modelling: Some experiences from education and industrial applications,” *Simulation Modelling Practice and Theory*, vol. 33, pp. 45–67, 2013.
- [63] CHEN, E., *Fuel Cell Technology Handbook*. CRC Press, 2003.
- [64] SPRINGER, T. E., ZAWODZINSKI, T. A., and GOTTESFELD, S., “Polymer electrolyte fuel cell model,” *Journal of the Electrochemical Society*, vol. 138, no. 8, pp. 2334–2342, 1991.
- [65] SPRINGER, T. E., WILSON, M. S., and GOTTESFELD, S., “Modeling and experimental diagnostics in polymer electrolyte fuel cells,” *Journal of the Electrochemical Society*, vol. 140, no. 12, pp. 3513–3526, 1993.
- [66] BERNARDI, D. M. and VERBRUGGE, M. W., “Mathematical model of a gas diffusion electrode bonded to a polymer electrolyte,” *American Institute of Chemical Engineers Journal*, vol. 37, no. 8, pp. 1151–1163, 1991.
- [67] BERNARDI, D. M. and VERBRUGGE, M. W., “A mathematical model of the solid polymer electrolyte fuel cell,” *Journal of the Electrochemical Society*, vol. 139, no. 9, pp. 2477–2491, 1992.
- [68] WEBER, A. Z. and NEWMAN, J. S., “Modeling transport in polymer-electrolyte fuel cells,” *Chemical Reviews*, vol. 104, no. 10, pp. 4679–4726, 2004.
- [69] MENNOLA, T., NOPONEN, M., ARONNIEMI, M., HOTTINEN, T., MIKKOLA, M., HIMANEN, O., and LUND, P., “Mass transport in the cathode of a free-breathing polymer electrolyte membrane fuel cell,” *Journal of Applied Electrochemistry*, vol. 33, pp. 979–987, Nov. 2003.
- [70] JEMEI, S., HISSEL, D., PERA, M., and KAUFFMANN, J., “On-board fuel cell power supply modeling on the basis of neural network methodology,” *Journal of Power Sources*, vol. 124, pp. 479–486, Nov. 2003.
- [71] LEE, W.-Y., PARK, G.-G., YANG, T.-H., YOON, Y.-G., and KIM, C.-S., “Empirical modeling of polymer electrolyte membrane fuel cell performance using artificial neural networks,” vol. 29, pp. 961–966, 2004.
- [72] WANG, C.-Y., “Fundamental models for fuel cell engineering,” *Chemical Reviews*, vol. 104, no. 10, pp. 4727–4765, 2004.
- [73] HARALDSSON, K. and WIPKE, K., “Evaluating PEM fuel cell system models,” *Journal of Power Sources*, vol. 126, pp. 88–97, 2004.

- [74] DJILALI, N., "Computational modelling of polymer electrolyte membrane (PEM) fuel cells: Challenges and opportunities," *Energy*, vol. 32, no. 4, pp. 269–280, 2007.
- [75] DAWES, J. E., HANSPAL, N. S., FAMILY, O. A., and TURAN, A., "Three-dimensional CFD modelling of PEM fuel cells: An investigation into the effects of water flooding," *Chemical Engineering Science*, vol. 64, pp. 2781–2794, Jun. 2009.
- [76] GRATTON, D., "Reduced-order, trajectory piecewise-linear models for nonlinear computational fluid dynamics," Master's thesis, Massachusetts Institute of Technology, Jun. 2004.
- [77] BERNING, T. and DJILALI, N., "A 3D, multiphase, multicomponent model of the cathode and anode of a PEM fuel cell," *Journal of the Electrochemical Society*, vol. 150, pp. 1589–1598, Dec. 2003.
- [78] NGUYEN, P., BERNING, T., and DJILALI, N., "Computational model of a PEM fuel cell with serpentine gas flow channels," *Journal of Power Sources*, vol. 130, pp. 149–157, May 2004.
- [79] WANG, Y. and WANG, C.-Y., "Dynamics of polymer electrolyte fuel cells undergoing load changes," *Electrochimica Acta*, vol. 51, no. 19, pp. 3924–3933, 2006.
- [80] SLATTERY, J. C. and BIRD, R. B., "Calculation of the diffusion coefficient of dilute gases and of the self-diffusion coefficient of dense gases," *American Institute of Chemical Engineers Journal*, vol. 4, pp. 137–142, Jun. 1958.
- [81] MAZUMDER, S. and COLE, J. V., "Rigorous 3-D mathematical modeling of PEM fuel cells: I. model predictions without liquid water transport," *Journal of the Electrochemical Society*, vol. 150, no. 11, pp. A1510–A1517, 2003.
- [82] MAZUMDER, S. and COLE, J. V., "Rigorous 3-D mathematical modeling of PEM fuel cells II: Model predictions with liquid water transport," *Journal of the Electrochemical Society*, vol. 150, no. 11, pp. A1503–A1509, 2003.
- [83] SOUSA, T., MAMLOUK, M., SCOTT, K., and RANGEL, C., "Three dimensional model of a high temperature PEMFC: Study of the flow field effect on performance," *Fuel Cells*, vol. 12, no. 4, pp. 566 – 576, 2012.
- [84] DUTTA, S., SHIMPALEE, S., and VAN ZEE, J., "Three-dimensional numerical simulation of straight channel PEM fuel cells," *Journal of Applied Electrochemistry*, vol. 30, no. 2, pp. 135–146, 2000.
- [85] CHIPPAR, P. and JU, H., "Three-dimensional non-isothermal modeling of a phosphoric acid-doped polybenzimidazole (PBI) membrane fuel cell," *Solid State Ionics*, vol. 225, pp. 30 – 39, 2012.
- [86] UM, S., WANG, C.-Y., and CHEN, K. S., "Computational fluid dynamics modeling of proton exchange membrane fuel cells," *Journal of the Electrochemical Society*, vol. 147, no. 12, pp. 4485–4493, 2000.
- [87] UM, S. and WANG, C.-Y., "Three-dimensional analysis of transport and electrochemical reactions in polymer electrolyte fuel cells," *Journal of Power Sources*, vol. 125, no. 1, pp. 40–51, 2004.

- [88] SIVERTSEN, B. R. and DJILALI, N., “CFD-based modelling of proton exchange membrane fuel cells,” *Journal of Power Sources*, vol. 141, no. 1, pp. 65–78, 2005.
- [89] SHIMPALEE, S., OHASHI, M., VAN ZEE, J., ZIEGLER, C., STOECKMANN, C., SADELER, C., and HEBLING, C., “Experimental and numerical studies of portable PEMFC stack,” *Electrochimica Acta*, vol. 54, no. 10, pp. 2899–2911, 2009.
- [90] MENG, H. and WANG, C.-Y., “Electron transport in PEFCs,” *Journal of the Electrochemical Society*, vol. 151, no. 3, pp. A358–A367, 2004.
- [91] MENG, H. and WANG, C.-Y., “Multidimensional modelling of polymer electrolyte fuel cells under a current density boundary condition,” *Fuel Cells*, vol. 5, no. 4, pp. 455–462, 2005.
- [92] PARTHASARATHY, A., SRINIVASAN, S., APPLEBY, A., and MARTIN, C. R., “Temperature dependence of the electrode kinetics of oxygen reduction at the platinum/Nafion[®] interface: A microelectrode investigation,” *Journal of the Electrochemical Society*, vol. 139, no. 9, pp. 2530–2537, 1992.
- [93] FULLER, T. F. and NEWMAN, J. S., “Water and thermal management in solid-polymer-electrolyte fuel cells,” *Journal of the Electrochemical Society*, vol. 140, no. 5, pp. 1218–1225, 1993.
- [94] GURAU, V., LIU, H., and SADIK, K., “Two-dimensional model for proton exchange membrane fuel cells,” *Journal of the American Institute of Chemical Engineers*, vol. 44, pp. 2410–2422, Nov. 1998.
- [95] OBAYOPO, S. O., BELLO-OCHEDE, T., and MEYER, J. P., “Three-dimensional optimisation of a fuel gas channel of a proton exchange membrane fuel cell for maximum current density,” *International Journal of Energy Research*, vol. 37, no. 3, pp. 228–241, 2013.
- [96] ANSYS, INC., *ANSYS FLUENT 12.0 Fuel Cell Module Manual*. Canonsburg, PA, Apr. 2009.
- [97] CD-ADAPCO, “es-pemfc: The CFD expert system for proton exchange membrane fuel cells,” Melville, NY, Nov. 2009. Product brochure.
- [98] COMSOL, INC., “Batteries & fuel cells module.” <http://www.comsol.com/products/batteries-fuel-cells/>. Accessed Mar. 2011.
- [99] MCCAIN, B. A., STEFANOPOULOU, A. G., and BUTTS, K. R., “A study toward minimum spatial discretization of a fuel cell dynamics model,” in *Proceedings of the International Mechanical Engineering Congress and Exposition (IMECE2006)*, (Chicago, IL), ASME, Nov. 2006. IMECE2006-14509.
- [100] NATARAJAN, D. and NGUYEN, T. V., “A two-dimensional, two-phase, multicomponent, transient model for the cathode of a proton exchange membrane fuel cell using conventional gas distributors,” *Journal of the Electrochemical Society*, vol. 148, no. 12, pp. A1324–A1335, 2001.
- [101] NGUYEN, T. V. and WHITE, R. E., “Water and heat management model for proton-exchange-membrane fuel cells,” *Journal of the Electrochemical Society*, vol. 140, no. 8, pp. 2178–2186, 1993.

- [102] BEVERS, D., WÖHR, M., YASUDA, K., and OGURO, K., “Simulation of a polymer electrolyte fuel cell electrode,” *Journal of Applied Electrochemistry*, vol. 27, pp. 1254–1264, 1997.
- [103] THE MATHWORKS INC., “MATLAB & Simulink,” Jan. 2007. Version R2007A.
- [104] LMS, “Fuel cell stack model.” <http://www.emmeskay.com/products/fuel-cell-stack-model>. Accessed Sep. 2013.
- [105] THE MATHWORKS INC., “Fuel cell stack.” <http://www.mathworks.com/help/toolbox/phymod/powersys/ref/fuelcellstack.html>. Part of SimPowerSystems, accessed Mar. 2011.
- [106] PARK, J. and MIN, K., “A quasi-three-dimensional non-isothermal dynamic model of a high-temperature proton exchange membrane fuel cell,” *Journal of Power Sources*, vol. 216, pp. 152–161, 2012.
- [107] FRANCO, A. A., JALLUT, C., and MASCHKE, B., “A multi-scale dynamic mechanistic model for transient analysis of PEFCs,” in *Proceedings of the World Hydrogen Energy Conference (WHEC)*, (Lyon, France), Jun. 2006.
- [108] FRANCO, A. A., SCHOTT, P., JALLUT, C., and MASCHKE, B., “A multi-scale dynamic mechanistic model for the transient analysis of PEFCs,” *Fuel Cells*, vol. 7, no. 2, pp. 99–117, 2007.
- [109] PUKRUSHPAN, J. T., STEFANOPOULOU, A. G., and PENG, H., *Control of Fuel Cell Power Systems*. Advances in Industrial Control, Springer, 2004.
- [110] VAHIDI, A., STEFANOPOULOU, A. G., and PENG, H., “Model predictive control for starvation prevention in a hybrid fuel cell system,” in *Proceedings of the American Control Conference*, vol. 1, pp. 834–839, Jun.–Jul. 2004.
- [111] VAHIDI, A., STEFANOPOULOU, A. G., and PENG, H., “Current management in a hybrid fuel cell power system: A model-predictive control approach,” *IEEE Transactions on Control Systems Technology*, vol. 14, pp. 1047–1057, Nov. 2006.
- [112] MCKAY, D. A., OTT, W. T., and STEFANOPOULOU, A. G., “Modeling, parameter identification, and validation of water dynamics for a fuel cell stack,” in *Conference on Fuel Cell Science, Engineering and Technology*, (Orlando, FL), ASME, Nov. 2005. FUELCELL2005-81484.
- [113] CHEN, J., SIEGEL, J. B., STEFANOPOULOU, A. G., and WALDECKER, J. R., “Optimization of purge cycle for dead-ended anode fuel cell operation,” *International Journal of Hydrogen Energy*, vol. 38, no. 12, pp. 5092–5105, 2013.
- [114] STEINMANN, W. D. and TREFFINGER, P., “Simulation of fuel cell powered drive trains,” in *Modelica Workshop Proceedings*, (Lund, Sweden), pp. 153–158, Modelica Association, Oct. 2000.
- [115] TREFFINGER, P. and GOEDECKE, M., “Development of fuel cell powered drive trains with Modelica,” in *Proceedings of the 2nd International Modelica Conference*, (Oberpfaffenhofen, Germany), Mar. 2002.
- [116] RUBIO, M. A., URQUIA, A., GONZÁLEZ, L., GUINEA, D., and DORMIDO, S., “FuelCellLib: A Modelica library for modeling of fuel cells,” in *Proceedings of the 4th International Modelica Conference*, (Hamburg-Harburg, Germany), Modelica Association, Mar. 2005.

- [117] RUBIO, M. A., URQUIA, A., and DORMIDO, S., “Experimental validation of the FuelCellLib Modelica library,” (Christchurch, New Zealand), pp. 1278–1283, 2009.
- [118] RUBIO, M. A., URQUIA, A., and DORMIDO, S., “Dynamic modelling of PEM fuel cells using the FuelCellLib Modelica library,” *Mathematical and Computer Modelling of Dynamical Systems*, vol. 16, pp. 165–194, Jun. 2010.
- [119] DAVIES, K. L. and MOORE, R. M., “Object-oriented fuel cell model library,” *Electrochemical Society Transactions*, vol. 11, pp. 797–808, Oct. 2007.
- [120] DAVIES, K. L. and MOORE, R. M., “PEMFCSim: A fuel cell model library in Modelica,” in *Proceedings of the 31st Fuel Cell Seminar & Exposition*, (San Antonio, TX), Oct. 2007.
- [121] BLUNIER, B. and MIRAOU, A., “Modelling of fuel cells using multi-domain VHDL-AMS language,” *Journal of Power Sources*, vol. 177, no. 2, pp. 434–450, 2008.
- [122] GAO, F., BLUNIER, B., MIRAOU, A., and EL-MOUDNI, A., “Cell layer level generalized dynamic modeling of a PEMFC stack using VHDL-AMS language,” *International Journal of Hydrogen Energy*, vol. 34, no. 13, pp. 5498–5521, 2009.
- [123] SALOGNI, A. and COLONNA, P., “Modeling of solid oxide fuel cells for dynamic simulations of integrated systems,” *Applied Thermal Engineering*, vol. 30, no. 5, pp. 464–477, 2010.
- [124] YUAN, X.-Z., SONG, C., WANG, H., and ZHANG, J., *Electrochemical Impedance Spectroscopy in PEM Fuel Cells: Fundamentals and Applications*. Springer-Verlag, 2009.
- [125] UNGETHÜM, J., “Fuel cell system modeling for real-time simulation,” in *Proceedings of the 4th International Modelica Conference*, (Hamburg-Harburg, Germany), Modelica Association, Mar. 2005.
- [126] MARINGANTI, R., KATRAGADDA, S., HAAS, R., and MAHALE, A., “A two-dimensional and transient model for a polymer electrolyte fuel cell cathode and membrane,” in *Proceedings of the European Simulation and Modelling Conference (ESM2005)* (TEIXEIRA, J. F. and BRITO, A., eds.), (Ostend, Belgium), pp. 406–410, Oct. 2005.
- [127] EBORN, J., PEDERSEN, L., HAUGSTETTER, C., and GHOSH, S., “System level dynamic modeling of fuel cell power plants,” in *Proceedings of the American Control Conference*, (Denver, CO), pp. 2024–2029, Jun. 2003.
- [128] SESHADRI, P. and KABIR, Z., “Steady state and transient modeling of a PEM fuel cell power plant for transportation applications,” in *Proceedings of ASME 3rd International Conference on Fuel Cell Science, Engineering and Technology*, (Ypsilanti, Michigan), May 2005. FUELCELL2005-74108.
- [129] ANDERSSON, D. and ÅBERG, E., “Dynamic modeling of a solid oxide fuel cell system in Modelica,” Master’s thesis, Lund University, Sweden, Feb. 2010.
- [130] ANDERSSON, D., ÅBERG, E., EBORN, J., YUAN, J., and SUNDÉN, B., “Dynamic modeling of a solid oxide fuel cell system in Modelica,” in *Proceedings of the 8th International Modelica Conference*, (Dresden, Germany), Modelica Association, Mar. 2011.
- [131] PROCESS SYSTEMS ENTERPRISE LIMITED, “gFuelCell.” <http://www.psenterprise.com/products/gfuelcell/index.html>. Accessed Sep. 2013.

- [132] MODELON AB, “Fuel cell library.” <http://www.modelon.com/products/modelica-libraries/fuel-cell-library>. Accessed Sep. 2013.
- [133] TAYLOR, B. N., *Guide for the Use of the International System of Units (SI): The Metric System*. Collingdale, PA: Diane Publishing Company, Nov. 1995.
- [134] CERAOLO, M., MIULLI, C., and POZIO, A., “Modelling static and dynamic behaviour of proton exchange membrane fuel cells on the basis of electro-chemical description,” *Journal of Power Sources*, vol. 113, no. 1, pp. 131–144, 2003.
- [135] KARNIK, A. Y., STEFANOPOULOU, A. G., and SUN, J., “Water equilibria and management using a two-volume model of a polymer electrolyte fuel cell,” *Journal of Power Sources*, vol. 164, no. 2, pp. 590–605, 2007.
- [136] KIM, G.-S., SUI, P. C., SHAH, A. A., and DJILALI, N., “Reduced-dimensional models for straight-channel proton exchange membrane fuel cells,” *Journal of Power Sources*, vol. 195, no. 10, pp. 3240–3249, 2010.
- [137] SPIEGEL, C., *PEM Fuel Cell Modeling and Simulation Using MATLAB*. Elsevier Academic Press, 2008.
- [138] SUNDÉN, B. and YUAN, J., “On modeling of heat and mass transfer and other transport phenomena in fuel cells,” *Frontiers in Heat and Mass Transfer*, vol. 1, no. 1, 2010.
- [139] WANG, Z. H., WANG, C.-Y., and CHEN, K. S., “Two-phase flow and transport in the air cathode of proton exchange membrane fuel cells,” *Journal of Power Sources*, vol. 94, no. 1, pp. 40–50, 2001.
- [140] WEBER, A. Z., DARLING, R. M., and NEWMAN, J. S., “Modeling two-phase behavior in PEFCs,” *Journal of the Electrochemical Society*, vol. 151, no. 10, pp. A1715–A1727, 2004.
- [141] YUAN, W., TANG, Y., PAN, M., LI, Z., and TANG, B., “Model prediction of effects of operating parameters on proton exchange membrane fuel cell performance,” *Renewable Energy*, vol. 35, pp. 656–666, Mar. 2010.
- [142] MCBRIDE, B. J., ZEHE, M. J., and GORDON, S., “NASA Glenn coefficients for calculating thermodynamic properties of individual species,” tech. rep., Glenn Research Center, Cleveland, OH, Sep. 2002. NASA/TP—2002-211556.
- [143] MORRISON, G., “The importance of including the liquid phase in equations of state for nonazeotropic refrigerant mixtures,” *ASHRAE Transactions*, vol. 91, no. 1B, pp. 260–273, 1985.
- [144] NAG, P. K., *Engineering Thermodynamics*. Tata McGraw-Hill, 4th ed., Apr. 2008.
- [145] MCGLASHAN, M. L., ed., *Chemical Thermodynamics*, vol. 1 of *Specialist Periodical Reports*. Royal Society of Chemistry, 1973.
- [146] DYMOND, J. H., MARSH, K. N., WILHOIT, R. C., and WONG, K. C., *Virial Coefficients of Pure Gases. Numerical Data and Functional Relationships in Science and Technology*, Springer-Verlag, 2002.
- [147] SALZMAN, W. R., “The virial expansion.” Course notes for *Physical Chemistry (Chemistry 480A)*, University of Arizona, Jul. 2004.

- [148] PRESENT, R. D., *Kinetic Theory of Gases*. McGraw-Hill, 1958.
- [149] RAO, Y. V. C., *Chemical Engineering Thermodynamics*. Hyderabad, India: Universities Press, 1997.
- [150] MORAN, M. J. and SHAPIRO, H. N., *Fundamentals of Engineering Thermodynamics*. John Wiley & Sons, 5th ed., 2004.
- [151] BEJAN, A., *Advanced Engineering Thermodynamics*. John Wiley & Sons, 3rd ed., 2006.
- [152] WOO, K. W. and YEO, S. I., "Dalton's law vs. Amagat's law for the mixture of real gases," *SNU Journal of Education Research*, vol. 5, pp. 127–134, 1995.
- [153] LI, D., SANKARAN, V., LINDAU, J. W., and MERKLE, C. L., "A unified computational formulation for multi-component and multi-phase flows," in *43rd Aerospace Sciences Meeting and Exhibit: Meeting Papers*, (Reno, NV), pp. 1041–1058, AIAA, 2005.
- [154] FLUENT INC., *FLUENT 6.3 User's Guide*. Lebanon, NH, Sep. 2006.
- [155] WESSELINGH, J. A. and KRISHNA, R., *Mass Transfer in Multicomponent Mixtures*. Delft, The Netherlands: Delft University Press, 2000.
- [156] KERKHOFF, P. J. A. M. and GEBOERS, M. A. M., "Toward a unified theory of isotropic molecular transport phenomena," *American Institute of Chemical Engineers Journal*, vol. 51, no. 1, pp. 79–121, 2005.
- [157] KERKHOFF, P. J. A. M. and GEBOERS, M. A. M., "Analysis and extension of the theory of multicomponent fluid diffusion," *Chemical Engineering Science*, vol. 60, no. 12, pp. 3129–3167, 2005.
- [158] KUROPATENKO, V. F., "Momentum and energy exchange in nonequilibrium multicomponent media," *Journal of Applied Mechanics and Technical Physics*, vol. 46, pp. 1–8, Jan. 2005.
- [159] MEIER, K., LAESECKE, A., and KABELAC, S., "Transport coefficients of the Lennard-Jones model fluid III: Bulk viscosity," *Journal of Chemical Physics*, vol. 122, no. 1, 2005.
- [160] BIRD, R. B., STEWART, W. E., and LIGHTFOOT, E. N., *Transport Phenomena*. John Wiley & Sons, revised 2nd ed., 2007.
- [161] YTREHUS, T., "Molecular-flow effects in evaporation and condensation at interfaces," *Multiphase Science and Technology*, vol. 9, no. 3, pp. 205–327, 1997.
- [162] BEDEAUX, D., KJELSTRUP, S., and RUBI, J., "Nonequilibrium translational effects in evaporation and condensation," *Journal of Chemical Physics*, vol. 119, no. 17, pp. 9163–9170, 2003.
- [163] CERCIGNANI, C., "Elementary solutions of the linearized gas-dynamics Boltzmann equation and their application to the slip-flow problem," *Annals of Physics*, vol. 20, pp. 219–233, 1962.
- [164] ASHCROFT, N. W. and MERMIN, N. D., *Solid State Physics*. New York: Holt, Rinehard and Winston, 1976.

- [165] RAH, K. and EU, B. C., "Analog of the Stokes-Einstein relation for bulk viscosity," *Physical Review Letters*, vol. 83, no. 22, pp. 4566–4569, 1999.
- [166] MAJUMDAR, P., *Computational Methods for Heat and Mass Transfer*. Series in Computational and Physical Processes in Mechanics and Thermal Sciences, Taylor & Francis, 2005.
- [167] BURGHARDT, A., "Mass transfer by diffusion," in *Encyclopedia of Life Support Systems (EOLSS)*, vol. I: Chemical Engineering and Chemical Process Technology, Oxford, UK: Developed under the Auspices of the UNESCO, EOLSS Publishers, 2013. Accessed Feb. 24, 2013.
- [168] BIRD, R. B., STEWART, W. E., and LIGHTFOOT, E. N., *Transport Phenomena*. John Wiley & Sons, 2nd ed., 2002.
- [169] LIGGETT, J. A., *Fluid Mechanics*. McGraw-Hill, 1994.
- [170] BOCKRIS, J. O. M., REDDY, A. K. N., and GAMBOA-ALDECO, M., *Modern Electrochemistry 2A: Fundamentals of Electrodeics*. Kluwer Academic/Plenum Publishers, 2nd ed., 2000.
- [171] TAYLOR, R. and KRISHNA, R., *Multicomponent Mass Transfer*. Wiley Series in Chemical Engineering, John Wiley & Sons, 1993.
- [172] INCROPERA, F. P. and DEWITT, D. P., *Fundamentals of Heat and Mass Transfer*. John Wiley & Sons, 5th ed., 2002.
- [173] MATUSZAK, D., ARANOVICH, G. L., and DONOHUE, M. D., "Thermodynamic driving force for molecular diffusion: Lattice density functional theory predictions," *Journal of Non-Equilibrium Thermodynamics*, vol. 31, no. 4, pp. 355–384, 2006.
- [174] KARIM, S. M. and ROSENHEAD, L., "The second coefficient of viscosity of liquids and gases," *Reviews of Modern Physics*, vol. 24, pp. 108–116, Apr. 1952.
- [175] SCHETZ, J. A. and FUHS, A. E., eds., *Handbook of Fluid Dynamics and Fluid Machinery*, vol. 1: Fundamentals of Fluid Dynamics. John Wiley & Sons, 1996.
- [176] LIENHARD, IV, J. H. and LIENHARD, V, J. H., *A Heat Transfer Textbook*. Cambridge, MA: Phlogiston Press, 3rd ed., Jan. 2006. Version 1.24.
- [177] ÇENGEL, Y. A. and CIMBALA, J. M., *Fluid Mechanics: Fundamentals and Applications*. McGraw-Hill, 2006.
- [178] SVEHLA, R. A., "Transport coefficients for the NASA Lewis chemical equilibrium program," tech. rep., Lewis Research Center, Cleveland, OH, Apr. 1995. NASA Technical Memorandum 4647.
- [179] MCBRIDE, B. J. and GORDON, S., "Computer program for calculating complex chemical equilibrium compositions and applications: II. Users manual and program description," NASA Reference Publication 1311, Lewis Research Center, Cleveland, OH, Jun. 1996. Recent data available at http://www.stanford.edu/~cantwell/AA283_Course_Material/CEAforPowerPCMac/CEAexec/trans.inp.

- [180] BOCKRIS, J. and NAGY, Z., “Symmetry factor and transfer coefficient: A source of confusion in electrode kinetics,” *Journal of Chemical Education*, vol. 50, pp. 839–843, Dec. 1973.
- [181] PRENTICE, G., *Electrochemical Engineering Principles*. International Series in the Physical and Chemical Engineering Sciences, Prentice Hall, Oct. 1991.
- [182] NEWMAN, J. S., *Electrochemical Systems*. International Series in the Chemical and Engineering Sciences, Prentice Hall, 2nd ed., 1991.
- [183] CELIA, M. A., RUSSEL, T. F., HERRERA, I., and EWING, R. E., “Eulerian-Lagrangian localized adjoint method for the advection-diffusion equation,” *Advances in Water Resources*, vol. 13, no. 4, pp. 187–206, 1990.
- [184] DEEN, W. M., *Analysis of Transport Phenomena*. Topics in Chemical Engineering, Oxford University Press, 1998.
- [185] TILLER, M., *Introduction to Physical Modeling with Modelica*. Springer, 2002.
- [186] DAVIES, K. L. and PAREDIS, C. J., “Natural unit representation in Modelica,” in *Proceedings of the 9th International Modelica Conference*, (Munich, Germany), Modelica Association, Sep. 2012.
- [187] BUREAU INTERNATIONAL DES POIDS ET MESURES, “The International System of Units (SI).” http://www.bipm.org/en/si/si_brochure/, Mar. 2006.
- [188] NATIONAL INSTITUTE OF SCIENCE AND TECHNOLOGY, “Fundamental physical constants: Complete listing.” <http://physics.nist.gov/cuu/Constants/Table/allascii.txt>, 2010. Accessed Jun. 2012.
- [189] McNISH, A. G., “Dimensions units and standards,” *Physics Today*, vol. 10, pp. 19–25, Apr. 1957.
- [190] WEST, D. B., *Introduction to Graph Theory*. Prentice Hall, 2nd ed., 2001.
- [191] WILLEMS, J. C., “Terminals and ports,” *IEEE Circuits and Systems Magazine*, vol. 10, no. 4, pp. 8–26, 2010.
- [192] BORUTZKY, W., *Bond Graph Modelling of Engineering Systems: Theory, Applications and Software Support*. Springer, 2011.
- [193] MODELICA ASSOCIATION, “Modelica: A unified object-oriented language for physical systems modeling: Language specification,” Mar. 2010. Version 3.2.
- [194] BUTT, H.-J., GRAF, K., and KAPPL, M., *Physics and Chemistry of Interfaces*. Wiley-VH Verlag, 2003.
- [195] PETZOLD, L. R., BRENNAN, K. E., and CAMPBELL, S. L., *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. North-Holland, Amsterdam: Society for Industrial Mathematics, 1989.
- [196] BINDER, W. R., PAREDIS, C. J. J., and GARCIA, H. E., “Hybrid energy system modeling in Modelica,” in *Proceedings of the 10th International Modelica Conference*, (Lund, Sweden), Modelica Association, Mar. 2014 (accepted).

- [197] KULIKOVSKY, A., DIVISEK, J., and KORNYSHEV, A., “Modeling the cathode compartment of polymer electrolyte fuel cells: Dead and active reaction zones,” *Journal of the Electrochemical Society*, vol. 146, no. 11, pp. 3981–3991, 1999.
- [198] WEBER, A. Z. and NEWMAN, J. S., “Modeling gas-phase flow in porous media,” *International Communications in Heat and Mass Transfer*, vol. 32, no. 7, pp. 855–860, 2005.
- [199] DAVIES, K. L., PAREDIS, C. J., and HAYNES, C. L., “Library for first-principle models of proton exchange membrane fuel cells in Modelica,” in *Proceedings of the 9th International Modelica Conference*, (Munich, Germany), Modelica Association, Sep. 2012.
- [200] DAVIES, K. L., MOORE, R. M., and BENDER, G., “Model library of polymer electrolyte membrane fuel cells for system hardware and control design,” in *Proceedings of the 7th International Modelica Conference* (CASELLA, F., ed.), (Como, Italy), Modelica Association, Linköping University Electronic Press, Sep. 2009.
- [201] DAVIES, K. L., HAYNES, C. L., and PAREDIS, C. J., “Modeling reaction and diffusion processes of fuel cells within Modelica,” in *Proceedings of the 7th International Modelica Conference* (CASELLA, F., ed.), (Como, Italy), Modelica Association, Linköping University Electronic Press, Sep. 2009.
- [202] HUGHES, D., FORD, J. C., DAVIES, K. L., HAYNES, C. L., WEPFER, W., and TUCKER, D., “A real-time spatial SOFC model for hardware-based simulation of hybrid systems,” in *Proceedings of the 9th Fuel Cell Science, Engineering and Technology Conference*, (Washington, DC), ASME, Aug. 2011.
- [203] BLOOMFIELD, J. P. and WILLIAMS, A. T., “An empirical liquid permeability: Gas permeability correlation for use in aquifer properties studies,” *Quarterly Journal of Engineering Geology and Hydrogeology*, vol. 28, pp. S143–S150, Nov. 1995.
- [204] CUSSLER, E. L., *Diffusion: Mass Transfer in Fluid Systems*. Cambridge University Press, 2nd ed., 1997.
- [205] BOUDIN, L., GREC, B., and SALVARANI, F., “A mathematical and numerical analysis of the Maxwell-Stefan diffusion equations.” www-dimat.unipv.it/~salvarani/BoudinGrecSalvarani.pdf, accessed Feb. 2011.
- [206] ROULSTON, D. J., *An Introduction to the Physics of Semiconductor Devices*. The Oxford Series in Electric and Computer Engineering, Oxford University Press, 1999.
- [207] YOUNG, H. D. and FREEDMAN, R. A., *University Physics*. Addison-Wesley Series in Physics, Addison-Wesley Publishing Company, 9th ed., Jun. 1996.
- [208] HOROWITZ, P. and HILL, W., *The Art of Electronics*. Cambridge University Press, 2nd ed., 1999.
- [209] SCHUMACHER, J. O. and WETTLING, W., *Device physics of silicon solar cells*, vol. 3 of *Photoconversion of Solar Energy*. London: Imperial College Press, Jun. 2001. ISBN 1-86094-161-3.
- [210] KUBO, R., “The fluctuation-dissipation theorem,” *Reports on Progress in Physics*, vol. 29, pp. 255–284, 1966.

- [211] DASSAULT SYSTÈMES, “Dymola: Dynamic Modeling Laboratory,” Mar. 2010. Version 7.4.
- [212] THATCHER, P. and SEIBERT, S., “html2latex.” <http://html2latex.sourceforge.net>, Sep. 2000. Perl script to convert HTML files to LaTeX files.
- [213] MOSES, B. and HEINZ, C., *The Listings Package*, Feb. 2007. Version 1.4.
- [214] TÖPEL, D., “Language extensions for syntax highlighting in the ‘Listings’ package,” Sep. 2009.
- [215] SPRY, D. B. and FAYER, M. D., “Proton transfer and proton concentrations in protonated Nafion fuel cell membranes,” *Journal of Physical Chemistry B*, vol. 113, no. 30, pp. 10210–10221, 2009.
- [216] LIN, J., LEE, J. K., KELLNER, M., WYCISK, R., and PINTAURO, P. N., “Nafion-flourinated ethylene-propylene resin membrane blends for direct methanol fuel cells,” *Journal of the Electrochemical Society*, vol. 153, no. 7, pp. A1325–A1331, 2006.
- [217] MARK, J. E., ed., *Polymer Data Handbook*. Oxford University Press, 1999.
- [218] SHAH, A. A., RALPH, T. R., and WALSH, F. C., “Modeling and simulation of the degradation of perfluorinated ion-exchange membranes in pem fuel cells,” *Journal of the Electrochemical Society*, vol. 156, no. 4, pp. B465–B484, 2009.
- [219] “Avogadro: An open-source molecular builder and visualization tool,” Apr. 2011. Version 1.03.
- [220] TAKENAKA, M. and MASUI, R., “Measurement of the thermal expansion of pure water in the temperature range 0 °C–85 °C,” *Metrologia*, vol. 27, no. 4, pp. 165–171, 1990.
- [221] SGL CARBON GROUP, “Sigracet[®] 10 series gas diffusion layer,” Apr. 2007. Doc. 04 2007/1 3NÄ.
- [222] MATSSON, S. E. and ELMQVIST, H., “Unit checking and quantity conservation,” in *Proceedings of the 6th International Modelica Conference*, (University of Applied Sciences, Bielefeld, Germany), Modelica Association, Mar. 2008.
- [223] BROMAN, D., ARONSSON, P., and FRITZSON, P., “Design considerations for dimensional inference and unit consistency checking in Modelica,” in *Proceedings of the 6th International Modelica Conference*, (Bielefeld, Germany), Modelica Association, Mar. 2008.
- [224] ARONSSON, P. and BROMAN, D., “Extendable physical unit checking with understandable error reporting,” in *Proceedings of the 7th International Modelica Conference*, (Como, Italy), Modelica Association, Sep. 2009.
- [225] NITTA, I., HIMANEN, O., and MIKKOLA, M., “Thermal conductivity and contact resistance of compressed gas diffusion layer of PEM fuel cell,” *Fuel Cells*, vol. 8, no. 2, pp. 111–119, 2008.
- [226] SGL CARBON GROUP, “Sigracet[®] 24 & 25 series gas diffusion layer,” Sep. 2004. Doc. DS FC 003 - GDL 24/25 - Rev00.
- [227] TORAY INDUSTRIES, INC., “Carbon paper.” <http://www.torayca.com/index2.html>, Sept. 2010.

- [228] ENTEGRIS, “Industrial graphite,” Apr. 2012.
- [229] DUPONT FUEL CELLS, “DuPontTM Nafion[®] PFSA membranes N-112, NE-1135, N-115, N-117, NE-1110,” Feb. 2004. Product specification.
- [230] KANDLIKAR, S. G. and LU, Z., “Thermal management issues in a PEMFC stack: A brief review of current status,” *Applied Thermal Engineering*, vol. 29, no. 7, pp. 1276–1280, 2009.
- [231] BROWN, W., WANG, P., PLIMPTON, S., and THARRINGTON, A., “Implementing molecular dynamics on hybrid high performance computers—short range forces,” *Computer Physics Communications*, vol. 182, no. 4, pp. 898–911, 2011.
- [232] HESS, B., KUTZNER, C., VAN DER SPOEL, D., and LINDAHL, E., “GROMACS 4: Algorithms for highly efficient, load-balanced, and scalable molecular simulation,” *Journal of Chemical Theory and Computation*, vol. 4, no. 3, pp. 435–447, 2008.
- [233] RAPAPORT, D. C., *The Art of Molecular Dynamics Simulation*. Cambridge University Press, 2nd ed., Apr. 2004.
- [234] GREINER, W., NEISE, L., and STÖCKER, H., *Thermodynamics and statistical mechanics*. Classical theoretical physics, Springer-Verlag, 1995.