

ABSTRACT

Title of Document: UNDERSTANDING DIRECT BOROHYDRIDE –
HYDROGEN PEROXIDE FUEL CELL
PERFORMANCE

Richard O'Neil Stroman,
Doctor of Philosophy, 2013

Directed By: Professor Gregory S. Jackson,
Department of Mechanical Engineering

Direct borohydride fuel cells (DBFCs) generate electrical power by oxidizing aqueous BH_4^- at the anode and reducing an oxidizer, like aqueous H_2O_2 for an all-liquid fuel cell, at the cathode. Interest in DBFCs has grown due to high theoretical energy densities of the reactants, yet DBFC technology faces challenges such as side reactions and other processes that reduce cell efficiency and power generation. Relationships linking performance to cell design and operation will benefit from detailed and calibrated cell design models, and this study presents the development and calibration of a 2D, single-cell DBFC model that includes transport in reactant channels and complex charge transfer reactions at each electrode.

Initial modeling was performed assuming ideal reactions without undesirable side reactions. Results were valuable for showing how design parameters impact ideal performance limits and DBFC cell voltage (efficiency). Model results showed that concentration boundary layers in the reactant flow channels limit power density and single-pass reactant utilization. Shallower channels and recirculation improve utilization, but at the expense of lower cell voltage and power per unit membrane

area. Reactant coulombic efficiency grows with decreasing inlet reactant concentration, reactant flow rate and cell potential, as the relative reaction rates at each electrode shift to favor charge transfer reactions.

To incorporate more realistic reaction mechanisms into the model, experiments in a single cell DBFC were performed to guide reaction mechanism selection by showing which processes were important to capture. Kinetic parameters for both electrochemical and critical heterogeneous reactions at each electrode were subsequently fitted to the measurements. Single-cell experiments showed that undesirable side reactions identified by gas production were reduced with lower reactant concentration and higher supporting electrolyte concentration and these results provided the basis for calibrating multi-step kinetic mechanism. Model results with the resulting calibrated mechanism showed that cell thermodynamic efficiency falls with cell voltage while coulombic utilization rises, yielding a maximum overall efficiency operating point. For this DBFC, maximum overall efficiency coincides with maximum power density, suggesting the existence of preferred operating point for a given geometry and operating conditions.

UNDERSTANDING DIRECT BOROHYDRIDE – HYDROGEN PEROXIDE
FUEL CELL PERFORMANCE

By

Richard O’Neil Stroman

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park, in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2013

Advisory Committee:

Professor Gregory S. Jackson, Chair

Professor Michael R. Zachariah

Assistant Professor Amir Riaz

Associate Professor Santiago de Jesus Solares

Associate Professor Chunsheng Wang, Dean’s Representative

Dedication

For Ina, Ingrid and Will.

Acknowledgements

This work would not have been possible without the generous support of many people. Family, friends and colleagues have given guidance, encouragement and assistance in innumerable ways. I cannot possibly thank them enthusiastically enough, but will try:

This project and my efforts to earn a graduate degree were generously funded by the Naval Research Laboratory Edison Memorial Training Program and the NRL Chemistry Division. This unique program is evidence of the high value placed on education at NRL, and a wonderful opportunity for those of us fortunate enough to work there. Additional funding for the latter part of this project was provided by Dr. Michelle Anderson at the Office of Naval Research, which was much appreciated.

I must thank my advisor, Prof. Greg Jackson, for guiding me through the PhD process and always being available to answer my questions or hash out new ideas, despite many competing pressures for his time. He sets an excellent example for those of us who seek to become better engineers by understanding more of the physical world.

Funding, while important, was the least of many contributions from NRL. It seems everyone there has offered advice, assistance, and at times an ear. In particular, I must thank Karen Swider-Lyons and my colleagues in the Alternative Energy Section for their guidance, assistance, encouragement and patience. They include Yannick Garsany, Ben Gould, Olga Baturina, Corey Love and Don Mase. Outside of Chemistry, Mike Schuette's help in the machine shop and elsewhere was much appreciated.

I must also express my gratitude to Chemistry Division management for their patience and continued support in this endeavor, including Barry Spargo, Warren Schultz and Rich Colton.

Finally, but far from least, is my family. My wife Ina has steadfastly supported me by holding down the home front, offering constant encouragement and her willingness to listen. Her tireless efforts to make time for me to complete this degree and her patience can be neither overstated nor overvalued. This could not have happened without her. My children Ingrid (3) and Will (1) have been amazingly forgiving while Dad worked every weekend, and they never allowed me to forget what was really important. My parents Neil and Mary Ann have encouraged and supported me for as long as I can remember, and more recently lent a hand so I could finish this. Thank you all.

Table of Contents

Dedication.....	ii
Acknowledgements.....	iii
Table of Contents.....	v
List of Tables.....	vii
List of Figures.....	viii
Nomenclature.....	xiv
Chapter 1: The Unrealized Promise of Direct Borohydride Fuel Cells.....	1
1.1 Introduction to Direct Borohydride Fuel Cells.....	1
1.2 DBFC Thermodynamics, Kinetics and Transport.....	6
1.2.1 Reaction Rates and the Activation Overpotential.....	12
1.2.2 Charge Balance and the Ohmic Overpotential.....	14
1.2.3 Transport and the Concentration Overpotential.....	16
1.2.4 Polarization and Power Curves as DBFC Performance Metrics.....	19
1.3 Alternative Cell Topologies and Chemistries.....	22
1.4 Prospective DBFC Benefits and Applications.....	25
1.5 Prior Work and the Present State of DBFC Research.....	26
1.5.1 DBFC Experiments with Aqueous Reactants.....	26
1.5.2 The Borohydride Oxidation Mechanism on Au.....	31
1.5.3 The Hydrogen Peroxide Reduction Mechanism on Pd:Ir.....	38
1.5.4 Additional DBFC Electrode Reactions.....	39
1.5.5 Rates for BH_4^- oxidation on Au and H_2O_2 reduction on Pd:Ir.....	41
1.5.6 Transport through Nafion Membranes in DBFCs.....	42
1.5.7 Mathematical Models of DBFCs and Similar Fuel Cells.....	45
1.6 The Present Study – A Step toward Understanding DBFC Performance.....	49
Chapter 2: Mathematical Model of a DBFC.....	52
2.1 Model Summary.....	52
2.1.1 Simplifications and Assumptions.....	52
2.1.2 Geometry and State Variables.....	56
2.1.3 Solution Approach.....	59
2.2 Governing Equations and Boundary Conditions.....	62
2.2.1 Pressure Residuals: Conservation of Mass.....	62
2.2.2 Mass Fraction Residuals: Conservation of Species.....	63
2.2.3 Velocity Residuals: Conservation of Momentum.....	64
2.2.4 Electric Potential Residuals: Electroneutrality.....	66
2.3 Species, Mass and Charge Fluxes in the Channels.....	67
2.4 Species, Mass and Charge Fluxes Through the Membrane.....	68
2.5 Species and Charge Fluxes at the Electrodes.....	71
2.6 Composition Equation of State.....	73
Chapter 3: An Analysis of Ideal DBFC Performance.....	74
3.1 Goals and Approach.....	74
3.2 Baseline Case.....	76
3.3 Effects of Reaction Rate Constant.....	83
3.4 Effects of Cell Geometry.....	85

3.5 Effects of Fuel Concentration and Flow Rate	89
3.6 Insights into DBFC Design from the Ideal Case Analysis	92
Chapter 4: Single-Cell DBFC Experiments.....	96
4.1 Goals and Approach	96
4.2 Development of a Single-Cell DBFC and Test Stand	97
4.2.1 Cell Hardware Design and Fabrication.....	97
4.2.2 Electro-deposition of Electrodes.....	99
4.2.3 Membrane Preparation.....	102
4.2.4 Test Stand Setup	102
4.3 Characterization of Electrodes.....	104
4.3.1 Optical Microscopy and Profilometry	104
4.3.2 SEM and EDS.....	106
4.3.3 Cyclic Voltammograms	110
4.4 Test Preparation and Procedures	112
4.4.1 Reactant Preparation.....	112
4.4.2 Measurement and Test Procedures	113
4.5 Results and Discussion	116
4.5.1 Measured Polarization Curves and Electrode Potentials.....	116
4.5.2 Gas Production Observations	128
4.5.3 Other Measurements and Observations	131
4.5.4 Conclusions.....	132
Chapter 5: Model-Based Analysis of a Realistic DBFC	135
5.1 Goals and Approach	135
5.2 Reaction Mechanism Fits to the Measurements	135
5.2.1 Fitting Approach and Procedure.....	137
5.2.2 Fitting the Simplest Reaction Mechanism.....	138
5.2.3 Fitting a Mechanism Including Cathode H^+ Reduction.....	144
5.3 Insights into Realistic DBFC Performance Provided by the Model.....	147
5.3.1 Realistic DBFC Performance	148
5.3.1 Influence of Fuel Flow Rate on Realistic Performance.....	158
5.3.2 Influence of Fuel BH_4^- Concentration on Realistic Performance	161
5.3.3 Conclusions from the Realistic DBFC Analysis	163
Chapter 6: Conclusions.....	165
6.1 Factors Governing DBFC Performance	165
6.1.1 Influence of Transport on Performance.....	166
6.1.2 Influence of Electrode Reactions on Performance	168
6.1.3 Recommendations for Improved DBFC Performance.....	169
6.2 Prospects for Practical DBFCs	171
6.3 The Utility and Limits of (Present) DBFC Models	172
6.4 Products of this Study.....	173
6.5 Recommendations for Future Work	174
Chapter 7: Appendices.....	176
7.1 DBFC Model Code.....	176
7.2 DBFC Model Calibration Code.....	302
Chapter 8: Bibliography	311

List of Tables

Table 1-1. Summary of DBFC experiments with aqueous $\text{NaBH}_4/\text{NaOH}$ fuel and H_2O_2 oxidizer	30
Table 2-1. KINSOL solver parameter values for the model	61
Table 2-2. Velocity boundary conditions in the model.....	66
Table 2-3. Diffusivities and apparent molar volumes of fuel and oxidizer solutes in H_2O at infinite dilution and 298 K.....	68
Table 2-4. Summary of thermophysical and transport properties in fully hydrated Nafion 115 in the Na^+ form at 298 K.....	70
Table 3-1. Baseline case and parameter variations	76
Table 3-2. Parameters common to all cases.....	76
Table 3-3. Performance metrics with respect to channel depth with all other parameters at the baseline case	86
Table 3-4. Performance metrics with respect to membrane thickness with all other parameters at the baseline case	88
Table 3-5. Performance metrics with respect to fuel flow rate with all other parameters at the baseline case	92
Table 4-1. Species abundances (%) in the cathode electrocatalyst strip, measured by EDS.....	108
Table 4-2. Density measurements and comparison to model predictions for fluids at 23°C. Fuel: 50 mM NaBH_4 / 2 M NaOH , Oxidizer: 250 mM H_2O_2 / 1 M H_2SO_4	132
Table 5-1. Model parameters for fitting process, taken from the experiments in Chapter 4. Both channels shared the same dimensions.	136
Table 5-2. Fitted reaction rate parameters assuming R 1.2 and R 1.4 at the anode and R 1.3 and R 1.5 at the cathode.	140
Table 5-3. Fitted reaction rate parameters assuming R 1.2 and R 1.4 at the anode and R 1.3, R 1.5 and R 1.23 at the cathode.....	146

List of Figures

Figure 1.1. Illustration of the DBFC cell configuration examined in this study, showing the cell geometry and ideal electrochemical processes taking place.	2
Figure 1.2. Schematic of a typical recirculated DBFC system.	5
Figure 1.3. Electrode and interface electric potentials in a DBFC. The orange line shows a typical electric potential distribution across the cell.	9
Figure 1.4. Example showing how the activation, ohmic and concentration overpotentials influence different regions of a fuel cell polarization curve.	20
Figure 1.5. Example power curve, showing the locations of the peak power operating point and desirable operating space.	21
Figure 1.6. Schematic illustrations of two DBFC cell topologies. Left, the catalyst layers are on the walls, which act as current collectors. Right, the catalyst layer is porous and located on the membrane. In both cases the reactant flows are perpendicular to the page.	23
Figure 1.7. Pourbaix diagram showing regions of stability, oxidation and reduction of H ₂ O as functions of pH and electrode potential [59].	40
Figure 1.8. Illustration of the electric potential profile through the membrane when the DBFC is operating (a) and a possible profile at open circuit (b), which would encourage crossover of other species.	44
Figure 2.1. Cell geometry and model domain with dimensions and spatial coordinates.	53
Figure 2.2. Activity of H ₂ O ₂ in water, predicted by empirical fit from [75].	56
Figure 2.3. Model domain in one <i>x</i> -direction slice across the cell, showing subdomains for channel cells, interfaces and electrodes. Only one channel cell is shown on each side for clarity.	57
Figure 2.4. Grid stencil showing relative locations of state variables. Cells for Y_k , ϕ and P are marked with dashed lines and one cell is shaded blue. One cell for v_x is shaded green and one cell for v_y is shaded red.	57
Figure 2.5. Typical channel discretization scheme near the fuel inlet (left).	58

Figure 2.6. Illustration of state variable locations and gradient calculation in the membrane sub-model.....	71
Figure 3.1. Baseline polarization curve showing activation, ohmic and concentration overpotentials.	77
Figure 3.2. Baseline power density vs. cell voltage. Points to the right of 2.2 V offer the best combination of power density and voltage efficiency.	78
Figure 3.3. Area specific resistance (ASR) associated with each overpotential at points down the channel, in the baseline case at 2.2 V.....	79
Figure 3.4. Baseline current density at 2.2 V at points down the channel, which correlates with the rate of reactant consumption. Anode and cathode current densities at each point are the same to within 0.01%.	80
Figure 3.5. Mole fluxes of BH_4^- in the y -direction, half way down the anode channel, as a profile from anode to membrane. Positive fluxes are toward the membrane and negative fluxes are toward the anode. Results for the baseline case at 2.2 V.	81
Figure 3.6. Contour plots of (a) BH_4^- , (b) BO_2^- and (c) Na^+ concentration ($\text{kmol}\cdot\text{m}^{-3}$) in the fuel channel for the baseline case at 2.2 V (peak power density). Channel boundaries are the same in each plot.	82
Figure 3.7. Polarization curves with $k_{RL,2,a} = 10^0, 10^3$ and $10^6 \text{ m}^4 \text{ mol}^{-1} \text{ s}^{-1}$ and $k_{RL,3,c} = 10^6 \text{ m}^4 \text{ mol}^{-1} \text{ s}^{-1}$. The bottom curve assumes k_a and k_c from Finkelstein et al. on Pt; anode: $2 \text{ m}^4 \cdot \text{kmol}^{-1} \cdot \text{s}^{-1}$ and cathode: $1.6 \text{ m}^4 \cdot \text{kmol}^{-1} \cdot \text{s}^{-1}$. All parameters, other than rate constants, are the same as in the baseline case, given in Table 3-1.	84
Figure 3.8. Area specific resistance at points down the channel, for the case with forward reaction rates set to 10^3 and cell voltage of 2.2 V. Aside from reaction rate constants, all parameters are the same as in the baseline case.....	85
Figure 3.9. Average power density vs. cell potential for channels 0.50, 0.75 and 1.00 mm deep. All other parameters as in the baseline case, labeled "BL". The bold dashed curve is drawn through the peak power density for each channel depth. The mean velocity of fuel solution entering the fuel channel was fixed at $0.1 \text{ m}\cdot\text{s}^{-1}$	87
Figure 3.10. Electric potential profiles across the cell from anode to cathode, near the inlet and near the outlet. The membrane clearly makes the largest contribution to the total ohmic losses, and the decrease in ohmic losses from inlet to outlet is related to the decrease in current density. Data in this figure are from the baseline at 2.2 V.....	88

Figure 3.11. Polarization and power density curves for membrane thicknesses of 1.45, 72.5, and 145 μm . All other parameters are the same as in the baseline case, which is labeled “BL”. The thickness of fully hydrated Nafion 115 in the Na^+ form is 145 μm [67].	89
Figure 3.12. Average power density vs. cell potential for fuel inlet concentrations ranging from 0.1 M to 0.5 M. All other parameters are the same as in the baseline case, labeled “BL”. The bold dashed curve is drawn through the maximum power density at each concentration.	91
Figure 3.13. Power density vs. cell voltage curves for fuel solution flow rates of 15, 30 and 60 $\text{mL}\cdot\text{min}^{-1}$. All other parameters are the same as in the baseline case, labeled “BL”. The bold dashed curve is drawn through the maximum power density at each flow rate.	92
Figure 3.14. Channel (single-pass) fuel utilization, overall fuel utilization, BH_4^- and BO_2^- concentrations at the inlet, all as functions of the fuel recirculation volume fraction. The concentration of BH_4^- in the fuel added to the recirculation loop was assumed to be 0.5 M. Aside from the inlet concentrations, all parameters are the same as in the baseline case.	95
Figure 4.1. Solid model of the DBFC used for calibration experiments. (A) Au-coated brass current collectors, (B) graphite electrodes, (C) PTFE channel masks, (D) Nafion 117 membrane, (E) PVDF reactant inlet and outlet.	98
Figure 4.2. Photograph of the DBFC used for calibration experiments.	99
Figure 4.3. Cathode (left) with Pd:Ir catalyst layer and anode (right) with Au catalyst layer.	101
Figure 4.4. Photograph of the setup for plating the DBFC anode.	101
Figure 4.5. Photograph of the test stand with DBFC.	103
Figure 4.6. Micrograph showing the boundary of the Au-coated and uncoated regions of the anode. Scale bar is 200 μm .	105
Figure 4.7. Electrode plated without first sanding the surface, which was dominated by artifacts of the plate machining process.	105
Figure 4.8. Anode with Au strip electro-deposited after sanding the graphite plate. Peaks in Au height correspond to the inlet during the first phase of electro-deposition.	106
Figure 4.9. Height profile across the middle (2.5 cm from each end) of the anode Au strip.	106

Figure 4.10. Example EDS spectrum from end “A” of the cathode electrocatalyst strip, showing relative abundances of each species.	107
Figure 4.11. SEM image of the Pd:Ir cathode prior to experiments, showing good coverage of the graphite plate.	109
Figure 4.12. SEM image of the Pd:Ir cathode prior to experiments, showing surface morphology consisting of rounded features.	109
Figure 4.13. SEM image of the Au anode prior to experiments, showing a different surface morphology with more texture than in the Pd:Ir cathode. The cathode image is inset with size adjusted to match the anode scale bar.	110
Figure 4.14. Cyclic voltammograms for the Au anode electrocatalyst and Pd:Ir cathode electrocatalyst. Measured in 0.5 M H ₂ SO ₄ with a scan rate of 20 mV·s ⁻¹	111
Figure 4.15. Polarization curves measured with 20 mM BH ₄ ⁻ / 2 M NaOH fuel and 40 mM H ₂ O ₂ / 1 M H ₂ SO ₄ oxidizer. One curve was generated by stepping from open circuit to 0.3 V, and the other by stepping from 0.3 V to open circuit.	114
Figure 4.16. Comparison of baseline (10 mM NaBH ₄ / 2 M NaOH fuel, 40 mM H ₂ O ₂ / 1 M H ₂ SO ₄ oxidizer) polarization curves, measured before and after experiments.	115
Figure 4.17. Measured polarization curves with varying BH ₄ ⁻ concentration in 2 M NaOH. In all cases, the oxidizer was 40 mM H ₂ O ₂ / 1 M H ₂ SO ₄ . Fuel and oxidizer flow rates are both 10 mL·min ⁻¹	118
Figure 4.18. Plots of measured current density vs. BH ₄ ⁻ concentration for the specified cell voltages.	119
Figure 4.19. Anode and cathode potentials measured during baseline polarization curve, vs. Ag/AgCl reference electrodes and then corrected to RHE.	121
Figure 4.20. Plots of the entire 10 mM and 20 mM BH ₄ ⁻ polarization curves.	123
Figure 4.21. 20 mM BH ₄ ⁻ polarization curves; “Original Membrane” is replotted here from the suite of five model calibration curves and “Fresh Membrane” is replotted here from the hysteresis experiment.	124
Figure 4.22. Measured power curves with varying BH ₄ ⁻ concentration in 2 M NaOH. In all cases, the oxidizer was 40 mM H ₂ O ₂ / 1 M H ₂ SO ₄ . Fuel and oxidizer flow rates were both 10 mL·min ⁻¹	126

Figure 4.23. Polarization and power curves measured for 50 mM NaBH ₄ / 2 M NaOH fuel and 250 mM H ₂ O ₂ / 1 M H ₂ SO ₄ oxidizer. Both flow rates were 10 mL·min ⁻¹ .	127
Figure 5.1. Comparison between measured and predicted polarization curves, with the model fitted to reaction mechanisms consisting of R 1.2 through R 1.4.	141
Figure 5.2. Predicted development of BH ₄ ⁻ and H ₂ O ₂ concentration boundary layers at the anode and cathode in the 20 mM BH ₄ ⁻ case. Solid lines are near the inlet and dashed lines are near the outlet.	142
Figure 5.3. Measured electrode potentials vs. RHE and predicted interfacial electrode potential differences Δφ for the 10 mM BH ₄ ⁻ baseline case.	143
Figure 5.4. Comparison between measured polarization curves and those predicted by the model, assuming R 1.2 through R 1.4 and R 1.23 occur.	145
Figure 5.5. Comparison between measured power curves and those predicted by the model, assuming R 1.2 through R 1.4 and R 1.23 occur.	147
Figure 5.6. Comparison between measured and predicted polarization curves for the 50 mM BH ₄ ⁻ / 250 mM H ₂ O ₂ case with 10 mL min ⁻¹ flow rates.	148
Figure 5.7. Predicted H ₂ (a) and O ₂ (b) concentrations in the channels for the baseline case at 1.1 V cell. Concentration units are mol·L ⁻¹ .	149
Figure 5.8. Predicted current density and H ₂ and O ₂ production rate distributions with respect to position in the channel, for the baseline case at 1.1 V cell.	150
Figure 5.9. Predicted coulombic efficiencies at the anode and cathode with respect to distance from the channel inlets, for the baseline case at 1.1 V cell.	151
Figure 5.10. Predicted power density and coulombic efficiency for 50mM BH ₄ ⁻ / 250mM H ₂ O ₂ with 10 mL·min ⁻¹ flow rates.	153
Figure 5.11. Plots of cell power density and coulombic utilization with respect to cell potential. Predictions for 50mM BH ₄ ⁻ /250mM H ₂ O ₂ with 10 mL·min ⁻¹ flow rates.	155
Figure 5.12. Electric potential profiles across the cell predicted by the calibrated model, at the midpoint (25 mm from the inlets), for the baseline case at 100 mA·cm ⁻² current density. The two curves compare results with R 1.4 and R 1.5 turned on and off.	156
Figure 5.13. Effective conversion efficiencies for fuel and oxidizer in the baseline case, plotted with power density for comparison of the peak locations.	157

Figure 5.14. Predicted BH_4^- concentration in the fuel channel for $10 \text{ mL}\cdot\text{min}^{-1}$ (a) and $1 \text{ mL}\cdot\text{min}^{-1}$ (b) flow rates, for the baseline case at 1.1 V. Both plots share the same color map, with concentrations in $\text{mol}\cdot\text{L}^{-1}$	158
Figure 5.15. Predicted power density for fuel flow rates from $1\text{-}15 \text{ mL}\cdot\text{min}^{-1}$, oxidizer flow rate of $10 \text{ mL}\cdot\text{min}^{-1}$ and $50\text{mM BH}_4^-/250\text{mM H}_2\text{O}_2$	159
Figure 5.16. Predicted coulombic efficiency (a) and coulombic utilization (b) for fuel flow rates from $1\text{-}15 \text{ mL}\cdot\text{min}^{-1}$, oxidizer flow rate of $10 \text{ mL}\cdot\text{min}^{-1}$, and $50\text{mM BH}_4^- / 250\text{mM H}_2\text{O}_2$	160
Figure 5.17. Predicted coulombic efficiency and utilization for BH_4^- concentrations from $10 - 70 \text{ mM}$, flow rates of $10 \text{ mL}\cdot\text{min}^{-1}$ and $250\text{mM H}_2\text{O}_2 / 1 \text{ M H}_2\text{SO}_4$ oxidizer.	163

Nomenclature

Symbol	Description	Units
μ_k	Chemical potential of species k	$\text{J}\cdot\text{kmol}^{-1}$
μ_k^0	Standard state chemical potential of species k	$\text{J}\cdot\text{kmol}^{-1}$
$\tilde{\mu}_k$	Electrochemical potential of species k	$\text{J}\cdot\text{kmol}^{-1}$
$\Delta\tilde{\mu}_{rxn}$	Net electrochemical potential change due to reaction	$\text{J}\cdot\text{kmol}^{-1}$
E_R^0	Standard half-cell potential of reaction R	V
E_{cell}^0	Standard cell potential	V
a_k	Activity of species k	none
C_k	Concentration of species k	$\text{kmol}\cdot\text{m}^{-3}$
T	Temperature	K
ϕ	Local electric potential	V
ϕ_a	Anode electric potential	V
ϕ_c	Cathode electric potential	V
$\phi_{a,int}$	Anode interface electric potential	V
$\phi_{c,int}$	Cathode interface electric potential	V
$\Delta\phi_a$	Electric potential difference between the anode and anode interface	V
$\Delta\phi_c$	Electric potential difference between the cathode and cathode interface	V
z_k	Charge of species k	none
$\eta_{act,a}$	Anode activation overpotential	V
$\eta_{act,c}$	Cathode activation overpotential	V
η_{ohmic}	Ohmic overpotential	V
$\eta_{conc,a}$	Anode concentration overpotential	V
$\eta_{conc,c}$	Cathode concentration overpotential	V
k_a	Anodic direction reaction rate constant for charge transfer reaction	varies
k_c	Cathodic direction reaction rate constant for charge transfer reaction	varies
k_f	Forward reaction rate constant for chemical reaction	varies

k_r	Reverse reaction rate constant for chemical reaction	varies
X_k	Mole fraction of species k	none
Y_k	Mass fraction of species k	none
P	Pressure	Pa
v_x	x -direction velocity	$\text{m}\cdot\text{s}^{-1}$
v_y	y -direction velocity	$\text{m}\cdot\text{s}^{-1}$
ρ	Mass density	$\text{kg}\cdot\text{m}^{-3}$
ρ_c	Net charge density	$\text{C}\cdot\text{m}^{-3}$
n_e	Number of electrons exchanged in charge transfer reaction	none
a_k	Activity of species k	none
R_{ch}^P	Residual for mass conservation in the channel; associated with pressure	$\text{kg}\cdot\text{s}^{-1}$
$R_{m_i}^P$	Residual for mass conservation at the membrane interface; associated with pressure	$\text{kg}\cdot\text{m}^{-2}\cdot\text{s}^{-1}$
$R_{k,ch}^{MF}$	Residual for species conservation in the channel; associated with mass fractions	$\text{kmol}\cdot\text{s}^{-1}$
R_{k,m_i}^{MF}	Residual for species conservation at the membrane interface; associated with mass fractions	$\text{kmol}\cdot\text{m}^{-2}\cdot\text{s}^{-1}$
R_{k,e_i}^{MF}	Residual for species conservation at the electrode interface; associated with mass fractions	$\text{kmol}\cdot\text{m}^{-2}\cdot\text{s}^{-1}$
R_{ch}^{xM}	Residual for x -direction momentum conservation in the channel; associated with x -direction velocity	$\text{m}\cdot\text{s}^{-2}$
R_{ch}^{yM}	Residual for y -direction momentum conservation in the channel; associated with y -direction velocity	$\text{m}\cdot\text{s}^{-2}$
R^ϕ	Residual for electroneutrality in the channels; associated with electric potential	$\text{C}\cdot\text{m}^{-3}$
R_k^{SF}	Residual for surface site fraction of species k	$\text{kmol}\cdot\text{m}^{-2}\cdot\text{s}^{-1}$
\vec{j}	Mass flux	$\text{kg}\cdot\text{m}^{-2}\cdot\text{s}^{-1}$
\vec{j}_k	Species (mole) flux	$\text{kmol}\cdot\text{m}^{-2}\cdot\text{s}^{-1}$
μ	Dynamic viscosity	$\text{Pa}\cdot\text{s}$
n_{d,Na^+}	Electro-osmotic drag coefficient for Na^+ in Nafion	$\frac{\text{kmol H}_2\text{O}}{\text{kmol}^{-1} \text{Na}^+}$

p_{H_2O}	Permeability of Nafion 115 to water	$m \cdot Pa^{-1} \cdot s^{-1}$
u_k	Mobility of species k	$m^2 \cdot V^{-1} \cdot s^{-1}$
f	F/RT	V^{-1}
F	Faraday's constant; 9.6485×10^7	$C \cdot kmol^{-1}$
\bar{R}	Ideal gas constant; 8.314×10^3	$J \cdot kmol^{-1} \cdot K^{-1}$
J	Jacobian matrix of the DBFC model	
\dot{V}	Volumetric flow rate	$m^3 \cdot s^{-1}$
ζ	Recirculation volume fraction	none
η_{ru}	Single-pass reactant utilization	none
η_{ce}	Coulombic efficiency (local)	none
η_{cu}	Coulombic utilization	none
I_{cell}	Cell current	A
i	Local current density	$A \cdot m^{-2}$
t	Time	varies
V_{cell}	Cell electric potential ($\phi_c - \phi_a$)	V
$\nu_{k,R}$	Stoichiometric coefficient for species k in reaction R	none
β_a, β_c	Anodic and cathodic direction symmetry factors in charge transfer reaction	none
$\Delta\phi'$	Equilibrium electrode-solution electric potential difference	V
K_{eq}	Reaction equilibrium constant	none
i_0	Exchange current density	$A \cdot m^{-2}$
D_k	Binary diffusivity of species k	$m^2 \cdot s^{-1}$
σ	Electrical conductivity	$S \cdot m^{-2}$
u_k	Mobility of species k	$m^2 \cdot V^{-1} \cdot s^{-1}$
γ_k	Activity coefficient for species k	None
I	Ionic strength of electrolyte solution	$kmol \cdot m^{-3}$
λ	Nafion membrane hydration	$kmol H_2O$ $kmol^{-1} SO_3^-$

Chapter 1: The Unrealized Promise of Direct Borohydride Fuel Cells

Direct borohydride fuel cells (DBFCs) generate electrical power by oxidizing aqueous BH_4^- (borohydride) and reducing aqueous H_2O_2 or gaseous O_2 . Interest in DBFCs has grown over the past decade due to several prospective advantages over batteries and other fuel cells. However, the technology remains immature and challenges have inhibited its use in practical applications. This chapter describes the present understanding of DBFCs, the prospective advantages and remaining challenges, and the ways this study aims to advance the technology toward practicality.

1.1 *Introduction to Direct Borohydride Fuel Cells*

Many different DBFC cell configurations have been tested in the literature. This study presents a configuration with electrodes separated from the ionic membrane by liquid reactant flow channels. The DBFC cell geometry and dominant electrochemical processes in this study are illustrated by Figure 1.1. DBFC operating principles are explained here in the context of this configuration. Alternative configurations are reviewed in a later section.

The cell used in this study consists of two parallel rectangular flow channels which are separated by a cation exchange membrane and bounded by walls. The walls constraining the flows function as current collectors. The fuel channel carries

an aqueous solution containing NaBH_4 and NaOH , and in this study, the oxidizer channel carries an aqueous solution containing H_2O_2 and H_2SO_4 . Each channel wall opposite the membrane is coated with an electrocatalyst; BH_4^- oxidation takes place at the anode (fuel channel wall) and H_2O_2 reduction takes place at the cathode (oxidizer channel wall). This cell geometry was chosen because it represents a class of DBFCs having separated electrodes and membranes. Such DBFCs have been considered in earlier studies [1, 2] because they are simple to fabricate, resist precipitate accumulation, expose the full membrane area to the electrolyte solutions and offer the possibility that migration will aid reactant transport in the channels. Furthermore, this DBFC can be represented in 2D, as shown in Figure 1.1, provided that the channel side walls are inert (electrochemically inactive) and spaced widely enough to have negligible effect on the hydrodynamics of the reactant flows. The 2D representation is more straightforward to model.

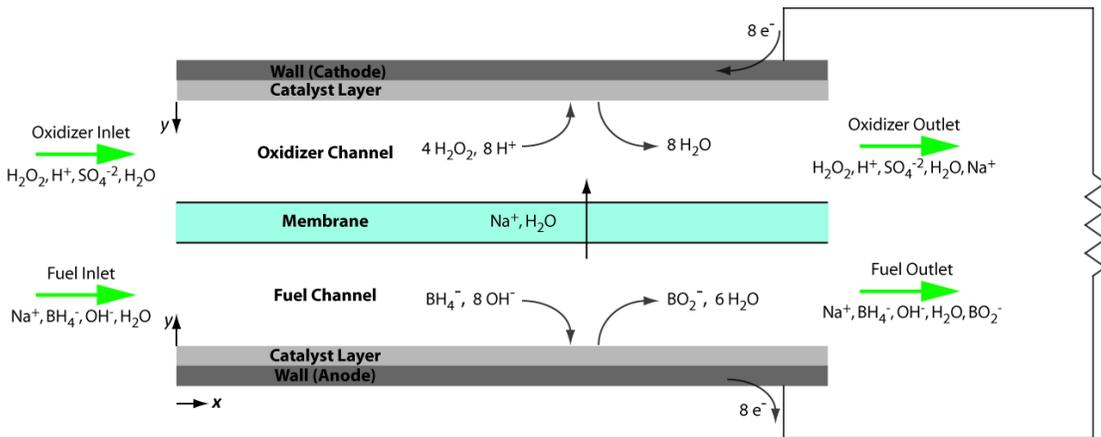


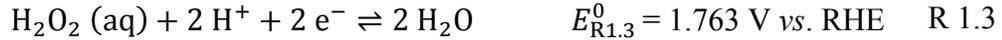
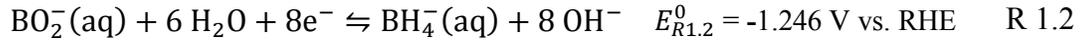
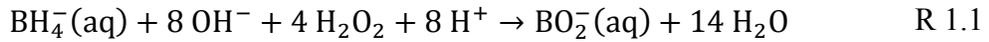
Figure 1.1. Illustration of the DBFC cell configuration examined in this study, showing the cell geometry and ideal electrochemical processes taking place.

Electrons provided to the anode by BH_4^- oxidation travel through an external circuit to the cathode to reduce H_2O_2 , while Na^+ ions maintain charge balance in the

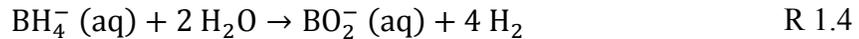
cell by crossing the membrane from fuel solution to oxidizer solution. The Na^+ ion flux through the membrane induces a H_2O flux due to electro-osmotic drag.

The net cell reaction consumes BH_4^- , H_2O_2 , OH^- , and H^+ and forms BO_2^- and H_2O . Eight electrons are liberated by the anode half-cell reaction (R 1.2) and travel through the external circuit to be consumed by the cathode half-cell reaction (R 1.3).

The standard cell potential for R 1.1 is $E_{\text{R1.1}}^0 = 3.01 \text{ V}$.



The fuel solution includes OH^- to provide a reactant for BH_4^- oxidation and stabilize the BH_4^- fuel, which is otherwise consumed by homogenous hydrolysis reaction R 1.4 to yield BO_2^- .



The rate of R 1.4 depends on pH and temperature; it occurs more rapidly at low pH. An empirical rate expression (Eq. 1.1) was developed by Kreevoy and Jacobson [3], which predicts the BH_4^- half-life (minutes) as a function of pH and temperature (K). According to Eq. 1.1, BH_4^- can be stored for months in high pH media ($\text{pH} \geq 13$).

$$\log(t_{1/2}) = \text{pH} - (0.034T - 1.92) \quad \text{Eq. 1.1}$$

Similarly, H^+ in the oxidizer solution provides a reactant for H_2O_2 reduction and stabilizes the H_2O_2 , which otherwise decomposes via R 1.5.



While R 1.4 and R 1.5 occur slowly in solutions having appropriate pH, the rates are nevertheless accelerated by contact with the anode and cathode catalysts. Heterogeneous catalysis of BH_4^- hydrolysis and H_2O_2 decomposition decreases the number of electrons per oxidized BH_4^- anion to less than 8 and the number of electrons per reduced H_2O_2 molecule to less than 2. The coulombic efficiency of each half cell reaction is often characterized by proximity to the theoretical number of electrons transferred; an anode which captures 4 electrons per consumed BH_4^- anion would have a coulombic efficiency of 50%.

DBFCs with the cell configuration presented above differ from common low-temperature fuel cells such as the proton exchange membrane fuel cell (PEMFC) and direct methanol fuel cell (DMFC). These differences stem from the use of aqueous electrolytes for the fuel and oxidizer, in contrast to the gaseous reactants in a PEMFC and the non-electrolyte aqueous fuel solution in a DMFC. The catalysts in PEMFCs and DMFCs are in contact with the ion-conducting membrane to enable H^+ participation in the electrochemical reactions, but the catalyst in a DBFC can be located elsewhere because the aqueous electrolytes in each channel support ion transport. Channel transport in PEMFCs and DMFCs is governed by convection and diffusion, but in a DBFC migration also contributes to (or inhibits) species transport.

Finally, slower diffusion in the liquid phase favors development of steeper concentration gradients and in a DBFC, accentuating down-the-channel effects on cell performance.

Flow-through DBFC topologies as in Figure 1.1 tend to have low channel reactant utilization rates; only a small fraction of the reactants flowing through each channel participates in the electrochemical reactions. The reactants are often recirculated to improve the utilization rates, as shown in Figure 1.2. Concentrated reactants are added to the recirculation loops upstream of the cell, and a portion of the effluent downstream of the cell is removed as waste. Since separation of reactants and products in the effluent streams is impractical, the cell operating conditions must be chosen to minimize the concentration of unreacted fuel in the effluent stream; otherwise fuel is lost to the waste tank.

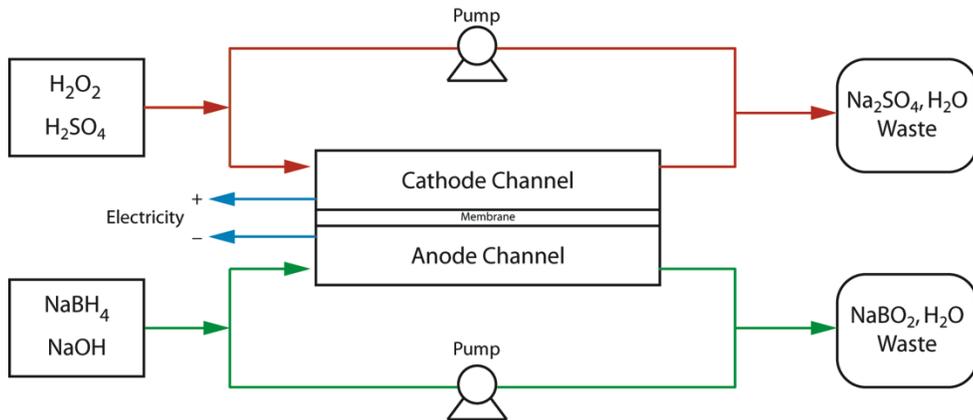


Figure 1.2. Schematic of a typical recirculated DBFC system.

Alternatively, the DBFC can be operated as a “flow battery” with a single reservoir in each loop. In the flow battery configuration, the entire contents of each

loop reservoir are recirculated until low reactant concentrations or high product concentrations inhibit cell performance.

1.2 DBFC Thermodynamics, Kinetics and Transport

The operating principle behind DBFCs, as in all fuel cells, is the conversion of chemical to electrical energy by electrochemical reactions occurring at the two electrodes. The available work that can be extracted from the chemical reactants in an electrochemical cell at a constant temperature and pressure is cast in terms of the change of Gibbs free energy ΔG of the global reaction. The chemical potential μ_k , equal to the intensive Gibbs free energy, is defined as the chemical potential at a standard reference state plus a correction for activity $a_k \neq 1$.

$$\mu_k = \mu_k^0 + RT \ln a_k \quad \text{Eq. 1.2}$$

When the reactants are charged species, though, electrochemical potential $\tilde{\mu}_k$ is a more appropriate description because it includes electrostatic potential energy. The electrochemical potential of species k is given by Eq. 1.3, in which z_k is the charge of species k , F is Faraday's constant and ϕ is the electric potential.

$$\tilde{\mu}_k = \mu_k + z_k F \phi \quad \text{Eq. 1.3}$$

For electrons, the only relevant energy is electrostatic and Eq. 1.3 simplifies to $\tilde{\mu}_e = -F\phi$. The energy $\Delta\tilde{\mu}_{R1.1}$ made available by the global fuel cell reaction is the difference between the electrochemical potentials of the reactants and products. With ν_k the stoichiometric coefficients in R 1.1, $\Delta\tilde{\mu}_{R1.1}$ can be written as:

$$\Delta\tilde{\mu}_{R1.1} = \sum_k \nu_{k,R1.1} \tilde{\mu}_k \quad \text{Eq. 1.4}$$

Similarly, the energy made available by each of the half-cell reactions is given by the difference in electrochemical potential of the half-cell reactants and products. Conservation of energy dictates that the total energy from the cell is equal to the sum of energies made available by the half-cell reactions.

$$\Delta\tilde{\mu}_{R1.1} = \Delta\tilde{\mu}_{R1.2} + \Delta\tilde{\mu}_{R1.3} \quad \text{Eq. 1.5}$$

The half-cell reactions cannot run independently; they are linked by an (effective) electroneutrality constraint and charge conservation. Electroneutrality is a result of the relationship between electric potential and local net charge density ρ_c described by Poisson's electrostatic equation (Eq. 1.6). The permittivity of free space ($\epsilon_0 = 8.85419 \times 10^{-12} \text{ F}\cdot\text{m}^{-1}$) is small, so small deviations from electroneutrality produce large electric potential gradients. The electric potential gradients tend to oppose charge stratification and drive charged species back into electrostatic equilibrium.

$$\nabla^2 \phi = -\frac{\rho_c}{\epsilon \epsilon_0} \quad \text{Eq. 1.6}$$

Significant deviations from ρ_c rarely occur outside the electrochemical double layers at electrode interfaces, where large values of $\nabla^2 \phi$ are found. Since the electrolyte solutions (and membrane) are electrically neutral, the net charge flux at the anode must be equal to the net charge flux at the cathode. However, local rates of reaction

can differ so long as the integrated rates over the entire area of the two electrodes are equal.

The same principles link cell voltage V_{cell} to the half-cell potentials $\Delta\phi_a$ and $\Delta\phi_c$, but before exploring that relationship, it will be helpful to clearly define the electric potential in the vicinity of each electrode. Electrodes in aqueous solution having electric potentials other than the point of zero charge (PZC) accumulate a layer of adsorbed water molecules due to the polar nature of H_2O . Ions may also accumulate near the electrode as their charges interact with the local electric field. The result is an accumulation of charge which produces a smooth transition over a short (~ 1 nm to 1 μ m) length scale from the electric potential in solution to the electric potential of the electrode [4].

Half-cell potentials are most often measured with respect to a reference electrode (RE), which has rapid reaction kinetics to maintain consistent potential despite changes in the local electrochemical environment. A common RE is the reversible hydrogen electrode (RHE), for which the reaction in low pH and neutral aqueous solutions is R 1.6. The standard reduction potential for R 1.6 is defined as 0.00 V.



Ideally the RE is situated in, or in electrochemical communication with, the region just outside the electrochemical double layer of the electrode. Doing so references the electrode potential to the electric potential just outside the double layer, and makes the electrode potential a measure of the potential drop across the double layer. If $\Delta\phi_a$ and $\Delta\phi_c$ are the electric potential drops across the anode and cathode double

layers, and $\phi_{a,int}$ and $\phi_{c,int}$ are the electric potential in the bulk solution just outside of each double layer, then the electrode potentials are: $\phi_a = \Delta\phi_a + \phi_{a,int}$ and $\phi_c = \Delta\phi_c + \phi_{c,int}$. The locations of ϕ_a , ϕ_c , $\Delta\phi_a$, $\Delta\phi_c$, $\phi_{a,int}$, and $\phi_{c,int}$ are shown in Figure 1.3, which adopts common conventions showing the change in electric potential across each double layer as if it were discontinuous and referencing all potentials in the system to the anode.

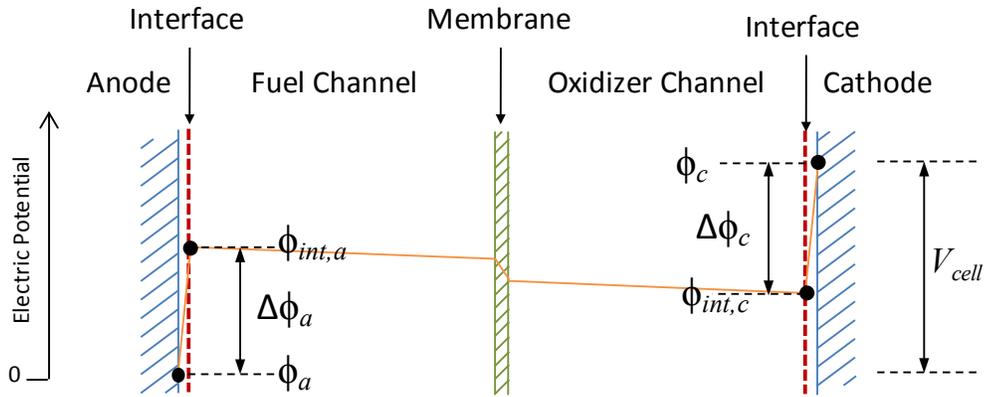


Figure 1.3. Electrode and interface electric potentials in a DBFC. The orange line shows a typical electric potential distribution across the cell.

The relationships between V_{cell} , $\Delta\phi_a$ and $\Delta\phi_c$ are most evident when the fuel cell is in the open circuit state and no current flows between the anode and cathode. At open circuit, R 1.2 and R 1.3 must each be at equilibrium so that no net electrons are transferred to/from the electrodes. Since the reactions are driven by the change in electrochemical potential, the equilibrium condition is:

$$\Delta\tilde{\mu}_{rxn} = \sum_k \nu_{k,R} \tilde{\mu}_{k,rxn} = 0 \quad \text{Eq. 1.7}$$

Substituting Eq. 1.3 for $\tilde{\mu}_{k,rxn}$ and solving for the interfacial electric potential difference yields the Nernst equation (Eq. 1.8).

$$\Delta\phi' = \frac{\Delta\mu_{rxn}^0}{n_e F} + \frac{RT}{n_e F} \ln \left(\prod a_k^{v_k} \right) \quad \text{Eq. 1.8}$$

The Nernst equation gives the electric potential difference between the aqueous solution and the electrode that will bring the reactants and products into equilibrium. $\Delta\phi'$ is equal to the standard half cell potential of the reaction E_{rxn}^0 under standard state conditions (i.e. $T = 298$ K and $a_k = 1$). The cell potential V_{cell} is the difference in electric potential between the anode and the cathode:

$$V_{cell} = \phi_c - \phi_a \quad \text{Eq. 1.9}$$

Since no net current flows through the cell at open circuit, the electric potentials at the electrode interfaces must be equal, because an electric potential gradient would drive net charge transfer. Thus at open circuit, $\phi_{a,int} = \phi_{c,int}$. Since the electrode potentials are referenced to the interface potentials, this makes the cell potential at open circuit equal to the sum of interfacial half-cell potentials, which is why:

$$E_{R1.1}^0 = E_{R1.2}^0 + E_{R1.3}^0, \quad \text{when} \quad \Delta\phi_a = E_{R1.2}^0 \quad \text{and} \quad \Delta\phi_c = E_{R1.3}^0.$$

In the $\text{BH}_4^- / \text{H}_2\text{O}_2$ DBFC examined in this study, the open circuit cell voltage (OCV) is $E_{R1.1}^0 = 1.763 \text{ V} - (-1.246 \text{ V}) = 3.01 \text{ V}$.

Fuel cell chemistries are chosen such that electrons delivered to the anode by the anode half-cell reaction have greater $\tilde{\mu}_e$ than electrons recovered from the cathode by the cathode half-cell reaction. When an electrically conductive path having non-zero

impedance (load) is provided, electrons are driven from anode to cathode by the difference in $\tilde{\mu}_e$, and the change in electron energy along this path is equal to the energy absorbed by the load. As the load impedance is decreased, the electrical current and electrochemical reaction rates increase to maintain balance between the electrochemical forces driving the reactions at each electrode. The higher rates of reaction incur greater internal cell losses, and the cell becomes less efficient as a smaller fraction of $\Delta\tilde{\mu}_{rxn}$ is available for the external load. The redistribution of $\Delta\tilde{\mu}_{rxn}$ in favor of internal cell losses at high current is manifested as a decrease in the cell voltage, often referred to as “overpotential”. The total overpotential is the sum of activation, ohmic and concentration overpotentials. Each overpotential is associated with a specific loss mechanism, as discussed in greater detail below. The cell potential at a given operating point is the open circuit cell potential minus all of the overpotentials at that operating point:

$$V_{cell} = E_{cell}^0 - \eta_{act,a} - \eta_{act,c} - \eta_{ohm} - \eta_{conc,a} - \eta_{conc,c} \quad \text{Eq. 1.10}$$

In general, as V_{cell} approaches OCV, the overpotentials are smaller and the fuel cell converts the reactants’ chemical energy into electrical energy more efficiently. A goal of most fuel cell research is to improve efficiency by decreasing the overpotentials. The sources of overpotential are discussed in detail in the next section, but broadly speaking, activation overpotentials can be minimized by choosing a more active catalyst, ohmic overpotentials can be minimized by raising the conductivity of electrolyte solutions and the membrane, and concentration overpotentials can be minimized by improving rates of transport to the electrodes.

1.2.1 Reaction Rates and the Activation Overpotential

An Arrhenius rate expression is often used to describe the rates of charge transfer reactions such as R 1.2 and R 1.3. In this case the net rate is the sum of forward and reverse rates (see Eq. 1.11). The direction supplying electrons to the electrode is “anodic” and the direction withdrawing electrons from the electrode is “cathodic”. The subscripts “a” and “c” in Eq. 1.11 refer to these directions.

$$r = k_a \prod_k a_{k,a}^{v_{k,a}} e^{n_e \beta_a f \Delta \phi} - k_c \prod_k a_{k,c}^{v_{k,c}} e^{-n_e \beta_c f \Delta \phi} \quad \text{Eq. 1.11}$$

The pre-exponential terms include rate constants and dependencies on the activities of species involved in the reaction. The exponential terms describe an activation energy barrier which depends on the magnitude of $\Delta\phi$, because the reaction must drive a net flux of charge through the double layer for the reaction to proceed. At the anode for example, R 1.2 must drive a net negative charge flux against the electric potential gradient in the double layer to the lower potential of the anode, doing work in the process. At the equilibrium interfacial electric potential difference predicted by the Nernst equation $\Delta\phi'$, the activation energy barrier magnitude is such that the anodic and cathodic rates are equal. For the reaction to run in the anodic direction (as it does at the anode of a functioning DBFC) $\Delta\phi_a$ must be less than $\Delta\phi'_a$. The shift in $\Delta\phi_a$ required for the reaction to proceed is the activation overpotential at the anode. In general, for both electrodes:

$$\eta_{act} = \Delta\phi - \Delta\phi' \quad \text{Eq. 1.12}$$

The symmetry factors β describe the relative slope of the activation energy barrier with respect to the reaction coordinate near equilibrium; $\beta_a = \beta_c = 0.5$ implies the barrier is symmetric. It is often assumed that $\beta_c = 1 - \beta_a$, i.e. the slope from each direction is nearly linear near equilibrium.

Eq. 1.11 can be recast in terms of overpotential to make the rate depend explicitly on the departure of $\Delta\phi$ from $\Delta\phi'$ [5]. First, recognize that the ratio of rate constants (equilibrium constant K_{eq}) depends on the magnitude of the activation energy barrier when the change in electrochemical potential due to charge transfer is zero, i.e. when $\Delta\phi = 0$. Then the activation energy barrier is the change in free energy of reaction ($\Delta\mu_{rxn}^0$) and the relationship between the anodic and cathodic rate constants can be written as in Eq. 1.13.

$$k_a/k_c = K_{eq} = e^{-\Delta\mu_{rxn}/RT} \quad \text{Eq. 1.13}$$

Then substitute Eq. 1.12 for $\Delta\phi$ and substitute Eq. 1.13 for k_c in Eq. 1.11. The rate constants and species dependencies can be collectively multiplied by Faraday's constant to give an exchange current density i_0 ; the remaining terms give the dependence on the overpotential. The result is the Butler-Volmer equation (Eq. 1.14).

$$i = i_0 \{ e^{\beta_a f \eta_{act}} - e^{-(1-\beta_a) f \eta_{act}} \} \quad \text{Eq. 1.14}$$

The exchange current density is the charge flux in each direction when the reaction is in (dynamic) equilibrium; the term in brackets biases the exchange current density in the anodic or cathodic direction depending on the value of η_{act} . When $\eta_{act} = 0$, the

anodic and cathodic rates are equal and no net current flows. The value of i_0 is often measured experimentally, but for this derivation it takes the form of Eq. 1.15, in which ν_e is the stoichiometric coefficient of the electrons. It is important to note that the exchange current density has additional dependencies on species concentration not given explicitly in Eq. 1.15; the change in free energy of reaction $\Delta\mu_{rxn}$ also depends on species concentrations, as shown by Eq. 1.2.

$$i_0 = \nu_e F k_a e^{\beta_a \Delta\mu_{rxn}/RT} \left(\prod_k a_{k,c}^{\nu_{k,c}} \right)^{\beta_a} \left(\prod_k a_{k,a}^{\nu_{k,a}} \right)^{(1-\beta_a)} \quad \text{Eq. 1.15}$$

1.2.2 Charge Balance and the Ohmic Overpotential

As discussed previously, the flow of electrons from anode to cathode must be balanced by a net positive ionic current through the cell in the same direction. The ionic current flows in response to electric potential gradients in the cell acting on the electric charge of ions; this transport process is called migration. The migration flux of species k is given by Eq. 1.16, where u_k is the mobility of species k .

$$\vec{J}_{mig,k} = -z_k u_k F C_k \nabla \phi \quad \text{Eq. 1.16}$$

The mobility is specific to the medium in which migration takes place, because it describes the amount of force which must be applied to the migrating ions by the electric field to overcome the opposition of interactions with the solvent. Mobility and diffusivity both describe relationships between a transport flux and the “force” (due to $\nabla\phi$ or ∇C) driving it. As such, they can be related for a given medium by the

Nernst-Einstein equation (Eq. 1.17). Eq. 1.17 can be used to obtain the mobility from the diffusivity or vice versa, when one parameter is unavailable.

$$u_k = \frac{D_k}{RT} \quad \text{Eq. 1.17}$$

The linear relationship between $\nabla\phi$ and $\vec{J}_{mig,k}$ in Eq. 1.16 is the origin of Ohm's Law in electrolyte solutions. The net charge flux, or current density, is

$$i = \sum_k \frac{\vec{J}_{mig,k}}{-z_k F}$$

which can be re-written as Ohm's Law (Eq. 1.18) if the conductivity is defined as $\sigma = \sum_k u_k C_k$.

$$i = \sigma \nabla\phi \quad \text{Eq. 1.18}$$

The cathode interface potential $\phi_{c,int}$ must be lower than the anode interface potential $\phi_{a,int}$ to produce the electric potential gradient necessary to drive the required charge flux (Figure 1.3), and that difference is the ohmic overpotential η_{ohm} (Eq. 1.19). The ohmic overpotential is manifested as a decrease in cell voltage as the changes in $\phi_{c,int}$ and $\phi_{a,int}$ bring ϕ_c and ϕ_a closer together. Together, Eq. 1.18 and Eq. 1.19 show that the ohmic overpotential is proportional to the current density.

$$\eta_{ohm} = \phi_{int,a} - \phi_{int,c} \quad \text{Eq. 1.19}$$

In PEM fuel cells the ohmic overpotential is almost entirely associated with the energy required to drive H^+ through the membrane. The DBFC in Figure 1.1 is similar in that the Na^+ flux through membrane dominates the ohmic overpotential, but

differs in that there are also contributions from the ionic resistivities of the fuel and oxidizer solutions in the channels. Ionic fluxes in the channels predominantly support the charge balancing Na^+ flux, but there are also migration fluxes of other ions to/from the electrodes, which support the electrochemical reactions. For example, the electric potential gradient drives cations (such as Na^+) toward the membrane as well as anions (such as BH_4^-) toward the anode. Driving BH_4^- toward the anode lowers the concentration and raises the low voltage of the anode. Thus the free energy available for electrical work is lowered. The voltage rise across the liquid reactants in the anode is modeled as part of the ohmic overpotential. The total ohmic overpotential is the sum of contributions from the fuel channel ($\eta_{ohm,f}$), membrane ($\eta_{ohm,m}$) and oxidizer channel ($\eta_{ohm,o}$).

1.2.3 Transport and the Concentration Overpotential

The net flux of each species k in the channels is the sum of contributions due to migration (when $z_k \neq 0$), diffusion and advection. This relationship is shown clearly by the Nernst-Planck equation (Eq. 1.20), in which the first term gives the migration and diffusion fluxes (driven by the gradient in electrochemical potential) and the second term gives the advection flux.

$$\vec{J}_k = -u_k \nabla \tilde{\mu}_k + \vec{v} C_k \quad \text{Eq. 1.20}$$

The diffusion and migration terms can be separated by substituting Eq. 1.3 for $\tilde{\mu}_k$, as in Eq. 1.21, and applying the Nernst-Einstein relation (Eq. 1.17) to cast diffusion in terms of the binary diffusivity D_k .

$$\vec{J}_k = -D_k \nabla \ln a_k - z_k u_k F C_k \nabla \phi + \vec{v} C_k \quad \text{Eq. 1.21}$$

To write the total flux in terms of concentration (a more convenient quantity than activity), the activity can be written as the product of concentration and a correction for non-ideal effects: $a_k = \gamma_k C_k$. The activity coefficient γ_k describes the deviation from ideal behavior; when species in solution do not interact, the solution is ideal and $\gamma_k = 1$. Recognizing that the activity coefficient is a function of concentration, the gradient in the diffusion term can be written as [6]:

$$\vec{J}_k = -D_k \left(1 + \frac{\partial \ln \gamma_k}{\partial \ln C_k} \right) \nabla C_k - z_k u_k F C_k \nabla \phi + \vec{v} C_k \quad \text{Eq. 1.22}$$

When the solution is ideal, the diffusion term simplifies to Fick's Law, and the Nernst-Planck equation becomes Eq. 1.23, which links the net flux of species k to the driving forces concentration gradient, electric potential gradient and bulk fluid velocity.

$$\vec{J}_k = -D_k \nabla C_k - z_k u_k F C_k \nabla \phi + \vec{v} C_k \quad \text{Eq. 1.23}$$

On all but very short timescales, and certainly at steady state, the species fluxes due to reactions taking place at the electrodes are matched by transport fluxes to/from the bulk solution in the channel. The rates of reaction (as discussed in §1.2.1) depend on ϕ and C_k at the electrode interface. The same is true for the species fluxes to/from the electrode interface; Eq. 1.23 shows that \vec{J}_k depends on ∇C_k and $\nabla \phi$. The necessity of matching the reaction fluxes and transport fluxes at the interface dictates the values of ϕ and C_k at the interface. For example, R 1.2 at the anode lowers the local BH_4^-

concentration by consuming BH_4^- . The lower BH_4^- concentration at the interface decreases the forward rate of reaction and increases the flux of BH_4^- to the interface from the bulk (by increasing $\nabla C_{\text{BH}_4^-}$). The electric potential at the interface shifts in a similar way to increase the migration flux. Together, the BH_4^- concentration and electric potential at the interface change until the reaction and transport fluxes match.

The fuel cell pays an efficiency penalty for the energy necessary to transport reactants to the electrodes and products away from the electrodes. The penalty appears as the concentration overpotential, arising from the changing concentrations at the electrode interfaces. Recall the Nernst equation (Eq. 1.8) gives the equilibrium value $\Delta\phi'$ at the electrode as a function of concentrations at the electrode interface. Near open circuit, $\Delta\phi' = E^0$, because the local concentrations are equal to the bulk concentrations. As the current density and rates of reaction are increased, local reactant concentrations fall and product concentrations rise. These changes depress $\Delta\phi'$ from the open circuit value at both electrodes, shrinking the difference between ϕ_c and ϕ_a . Since $V_{cell} = \phi_c - \phi_a$, the changes appear as a decrease in cell potential. The total loss of cell potential due to transport at each electrode is the sum of losses at the anode ($\eta_{conc,a}$) and cathode ($\eta_{conc,c}$).

The magnitude of the concentration overpotential is also affected by competition between the desired charge transfer reactions and other reactions taking place at the electrodes. For example, R 1.4 consumes BH_4^- and produces BO_2^- at the anode, which shifts the local concentrations and $\Delta\phi'$ without contributing to the current density. The loss of fuel to R 1.4 not only decreases coulombic efficiency by yielding

fewer electrons per BH_4^- consumed, but also decreases the cell potential at given current density. R 1.5 has a similar affect on the cathode.

Migration affects the concentration overpotential. Returning to the example of BH_4^- transport, when migration fluxes augment the total BH_4^- flux to the anode, an energy penalty is paid in the form of a larger ohmic overpotential. Conversely, the migration flux acts to *decrease* the concentration overpotential by raising the BH_4^- concentration at the anode for a given current density. The net result is (as shown in Chapter 3) greater peak power at the cost of lower efficiency at the peak power operating point.

1.2.4 Polarization and Power Curves as DBFC Performance Metrics

A common metric for describing fuel cell performance is the polarization curve, in which cell potential is plotted as a function of average cell current density. The term “polarization” originates in the electrochemistry community where it describes the electric potential difference between two phases; in this case, the fuel cell anode and cathode. Polarization curves contain a great deal of information about the activation, ohmic and concentration overpotentials in a fuel cell because each loss mechanism has a different dependence on current density. An example is shown in Figure 1.4, where the cell potential is defined by the OCV minus all three overpotentials, as in Eq. 1.10. All loss mechanisms are present at all operating points, but the relative magnitudes change depending on the value of the current density.

As the load impedance is decreased from the open circuit condition, ϕ_c approaches ϕ_a , so that the cell potential decreases and current density increases. The relationship between current density and cell potential is at first dominated by the

activation overpotential due to the exponential dependence of the reaction rates on the activation overpotentials. As the load impedance is further reduced, the forward exponential terms in the reaction rates become large enough that only small increases in activation overpotential are necessary to increase the current density, so the ohmic overpotential begins to dominate the relationship between current density and cell potential. Finally, as the current density and reaction rates grow large, transport limitations cause large changes in the concentrations near the electrodes and the concentration overpotential dominates.

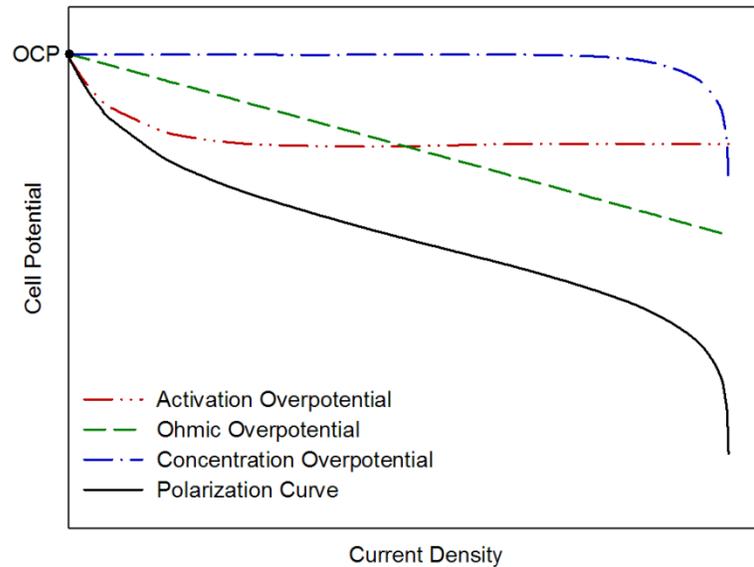


Figure 1.4. Example showing how the activation, ohmic and concentration overpotentials influence different regions of a fuel cell polarization curve.

The abrupt decrease in cell potential at high current density is an important feature of fuel cell performance called the transport limit. In most fuel cells, transport of uncharged reactants such as H_2 and O_2 to the electrodes occurs primarily through diffusion (perhaps enhanced by convection). As the current density increases, reactant concentrations near the electrodes fall, and at some value of the current

density (i_{lim}) the concentrations approach zero. No further increase in current density is possible because ∇C_k has attained the maximum value for a given set of operating conditions (inlet concentrations, flow rate, etc.). If the load impedance is further reduced, then the cell voltage continues to decrease with no concomitant increase in the current density; the result is a decline in power density (see Figure 1.5). Operating at any power density beyond the peak power point is undesirable because the same power density could be achieved at another point on the curve with lower current density and higher cell potential. Higher cell potential indicates the overpotentials are smaller at the alternative operating points, hence the cell operates more efficiently there.

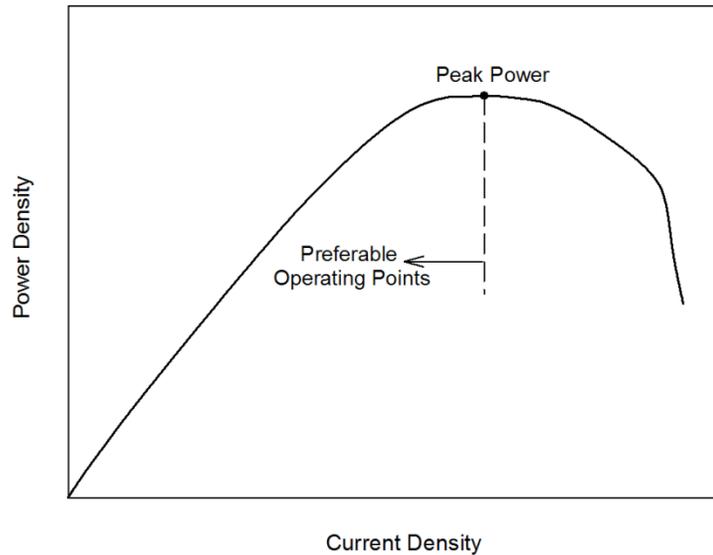


Figure 1.5. Example power curve, showing the locations of the peak power operating point and desirable operating space.

DBFCs with the geometry of Figure 1.1 can exhibit different transport limited behavior due to the influence of migration. When the rate of diffusion transport to the

electrodes is limited, the migration flux may continue to increase with the electric potential gradient. Decreases in cell potential increase the electric potential gradient, can therefore drive current densities beyond the diffusion limited current density. In these cases the current density increases as the cell potential decreases, until a limiting value is reached at short circuit (cell potential equal to zero). The significance of this effect depends on the mobilities and concentrations of the species involved, and the cell geometry.

1.3 Alternative Cell Topologies and Chemistries

The DBFC illustrated in Figure 1.1 is not the only DBFC topology to have been investigated. Experimental results have been reported for cells with a catalyst layer on the channel wall [1, 2, 7], catalyst coated on the membrane [8-13] and catalyst distributed throughout the channel on a porous medium [14, 15] (see Figure 1.6). In some cases the catalyst has been distributed on a porous medium (such as graphite felt paper [14, 16, 17], Ni foam [18], Ni mesh [12], Ti mesh [19]) situated between the flow channel and membrane. In some experiments the metallic catalyst consisted of particles mixed with Vulcan carbon, and in others the catalyst was electrodeposited directly onto the electrode. The relative strengths and weaknesses of the different electrocatalyst locations for both the anode and the cathode are unclear because of the lack of good models or direct experimental comparison.

In general, DBFC fuel solutions consist of aqueous BH_4^- and OH^- with an alkali metal cation. Experiments with Li^+ and K^+ have been reported [8, 20] however the most common cation is Na^+ . The rationale for selecting Na^+ is rarely discussed in the

literature, but it may be related to the use of Nafion membranes in the Na^+ form, which are readily available and relatively well understood.

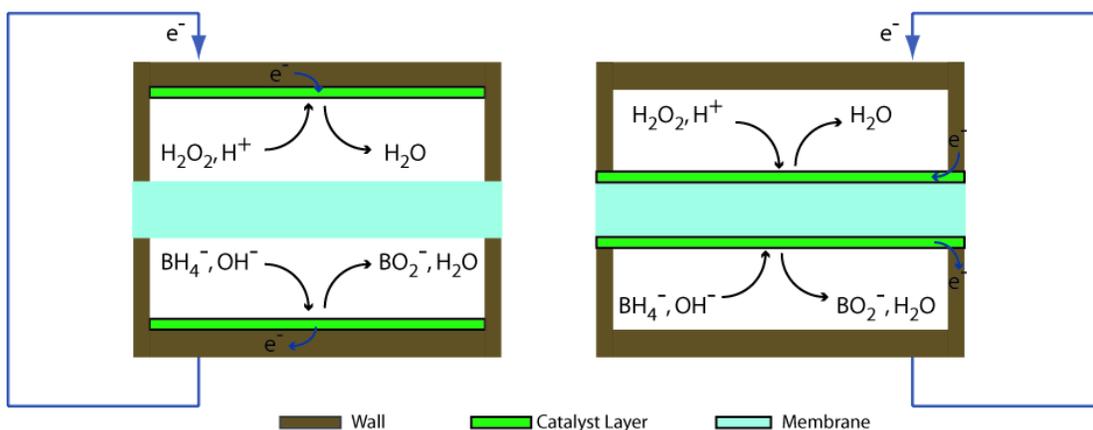
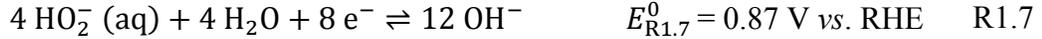


Figure 1.6. Schematic illustrations of two DBFC cell topologies. Left, the catalyst layers are on the walls, which act as current collectors. Right, the catalyst layer is porous and located on the membrane. In both cases the reactant flows are perpendicular to the page.

Experiments with acids other than H_2SO_4 have also been reported; for example, de Leon [16] reported experiments with H_2O_2 in HCl . Studies examining the affects of acid selection on the kinetics of H_2O_2 reduction have shown that some acid anions inhibit R 1.3, likely by adsorbing and blocking catalyst reaction sites [21]. Specifically, the activity of H_2O_2 reduction on Pd declines with acid anion in the order ClO_4^- , SO_4^- , Cl^- [21]. Acid selection also affects the rate of reaction R 1.5, likely also due to adsorption of anions on the catalyst surface. The rate of catalytic H_2O_2 decomposition has been shown to decrease with acid anion in the order HI , HBr , HCl , $\text{C}_2\text{H}_4\text{O}_2$, H_3PO_4 , H_2SO_4 , HClO_4 [22]. H_2SO_4 is often chosen because it offers reasonable rates for R 1.3 and R 1.5, insofar as the choice of acid can affect these rates.

While low pH oxidizer solutions may be preferable, they are not strictly necessary. Experiments with high pH oxidizer solutions have been reported [2, 23], in which the cathode reaction becomes R1.7. Alkaline oxidizer solutions yield lower cell voltage due to the lower standard reduction potential and tend to have slower kinetics [24], but they may provide some advantages over acidic oxidizer solutions. Among these are the opportunity to balance water fluxes through the membrane (discussed in §3.2) by shuttling fluid from the oxidizer recirculation loop to the fuel recirculation loop [2]. Another prospective advantage is the opportunity to use an anion exchange membrane, which can decrease OH⁻ storage requirements by moving OH⁻ produced by the cathode reaction to the anode, where it becomes a reactant for BH₄⁻ oxidation [2].



DBFC experiments with pure O₂ and humidified air have also been reported [8, 9]. In these experiments the cathode reaction becomes R1.8, for which the standard half cell potential is $E_{\text{R1.8}}^0 = 0.400 \text{ V vs. RHE}$. The standard reduction potential for R1.8 is 1.363 V lower than that of H₂O₂ reduction in acidic media, yielding substantially lower cell voltage. Furthermore, the kinetics for O₂ reduction are slower than for H₂O₂ [21], leading to larger activation overpotentials and lower efficiency.



1.4 *Prospective DBFC Benefits and Applications*

Interest in DBFCs has grown due to several desirable features of this technology. The reactants can be stored at sufficiently high concentrations to give theoretical energy densities greater than state-of-the-art rechargeable batteries¹ [25]. The aqueous reactant solutions can be stored at ambient temperature and pressure. The theoretical cell voltage for R 1.1 (3.01 V) substantially exceeds theoretical cell voltages of H₂-O₂ fuel cells (1.23 V). Moreover, the reactant chemical energy can be converted directly to electricity without any chemical preprocessing, thereby reducing power system complexity and improving reliability.

With respect to environmental concerns and sustainability, processes for reducing BO₂⁻ back to BH₄⁻ have been developed, so the possibility of a “closed” fuel cycle exists [26]. These electrochemical processes could be driven by a variety of renewable power sources, making the BH₄⁻/BO₂⁻ couple an energy carrier rather than simply a fuel [27]. While the possibility of such a fuel cycle has been demonstrated, technological immaturity (inefficiency, suboptimal process design, etc.) and a lack of infrastructure remain significant hurdles to practicality.

While challenges remain, continued progress has prompted interest in DBFCs for applications where energy density is important, particularly for air-independent propulsion [28, 29], remote sensors and portable electronics.

¹ Energy density depends on the assumed reactant concentrations and operating conditions. For example, in the case of 1 M NaBH₄ / 8 M NaOH fuel and 4 M H₂O₂ / 8 M H₂SO₄ oxidizer, and assuming the standard cell potential for reaction R 1.1, the theoretical energy density on a reactant basis is 322 W·hr·L⁻¹.

1.5 *Prior Work and the Present State of DBFC Research*

Recent interest in DBFCs began with a 1999 paper by Amendola et al. [8] which described a cell utilizing the direct oxidation of BH_4^- and reduction of atmospheric O_2 to produce electric power. The decade from 2000 to 2010 saw increasing energy demands by portable electronic devices and vehicles, and waning interest in DMFCs as a possible solution, due to problems with methanol crossover through the membrane [30-32]. The DBFC emerged as a potential alternative to the DMFC for portable power, and the prospect of air independent operation with H_2O_2 oxidizer led to further interest for underwater propulsion and space applications. The result was a rapid increase in the frequency of publication on all topics related to DBFCs, as documented in a review article by Merino-Jimenez et al. [25].

Many review papers [7, 9, 20, 24, 25, 27, 33, 34] have summarized DBFC research since 1999, in testament to the degree of interest in this technology and the remaining hurdles. A comprehensive review here would simply repeat efforts already undertaken in the literature; however a discussion of prior work directly relevant to this study is appropriate.

1.5.1 DBFC Experiments with Aqueous Reactants

While many papers describe cells utilizing O_2 as the oxidant, cells with liquid reactant streams at both the anode and cathode are more germane to the present study. Of these, the first was reported by Raman et al. [17] in 2004. The Raman DBFC consisted of rectangular flow channels cut into graphite plates, separated by a Nafion 117 membrane. The anode catalyst was a mixture of $\text{MmNi}_{3.55}\text{Al}_{0.3}\text{Mn}_{0.4}\text{Co}_{0.75}$ and Vulcan carbon, in which Mm refers to a Misch metal consisting of La (30 wt%), Ce

(50 wt%), Nd (15 wt%) and Pr (5 wt%). The cathode catalyst was Pt/C. Slurries containing the anode and cathode catalysts were deposited on carbon paper, which was then hot-pressed onto each side of the membrane. The fuel (10 wt% NaBH₄ in 20 wt% NaOH) and oxidizer (15% w/v H₂O₂ in varied concentrations of H₂SO₄) solutions were recirculated. This first Raman study demonstrated the feasibility of a NaBH₄ / H₂O₂ fuel cell and peak power densities of 120 mW·cm⁻² at 35°C and 350 mW cm⁻² at 70°C. More importantly, however, the Raman study characterized the rates of loss to R 1.4 and R 1.5 and identified the cathode as limiting the cell current density under the examined operating conditions. The rate of O₂ production was shown to increase with increasing oxidizer pH, as lower H⁺ concentrations favored R 1.5 over R 1.3. In situ reference electrodes were used to measure the potentials of the anode and cathode, and the cathode potential fell substantially as current density was increased, whereas the anode potential remained unchanged. The overall coulombic efficiency of the cell was reported as 83%, although the origin (R 1.4 or R 1.5) of the cathode losses was not given. Later studies by Raman et al. examined similar DBFCs, but with alkaline oxidizer [23] and a Nafion 961 membrane [35]. The alkaline oxidizer demonstrated lower cell potential as predicted by the standard reduction potential of R1.7. The Nafion 961 membrane yielded higher cell potential over long periods of recirculation due to reduced rates of OH⁻ crossover from the fuel solution (lower oxidizer pH). The polarization curves in [17] and [23] are dominated by the ohmic overpotential and lack clear activation and concentration regions shown in Figure 1.4. Polarization curves dominated by the ohmic overpotential are a common feature in DBFC experiments with high reactant concentrations.

The first of two studies published by de Leon et al. [16] examined the performance of a single DBFC and stacks of two and four cells. Commercially available filter-press cells, catalysts and diffusion media were used. The anode was $0.5 \text{ mg}\cdot\text{cm}^{-2}$ Au/C on carbon cloth and the cathode was $4.0 \text{ mg}\cdot\text{cm}^{-2}$ Pt/C on carbon paper. A Nafion 117 membrane separated the fuel and oxidizer compartments in each cell. The fuel (25% NaBH₄ in 6 M NaOH) and oxidizer (1 M H₂O₂ in 1 M HCl and 3 M NaCl) were recirculated throughout each experiment. Contrary to the findings of Raman, de Leon observed similar magnitude changes in the anode and cathode potentials with increasing current density, indicating that cell performance depends on operating conditions and/or choice of anode catalyst.

The second de Leon study [14] used the same filter-press cell, but with a Pd:Ir coated microfibrous cathode having high surface area. The Pd:Ir catalyst was electrodeposited on the cathode while circulating PdCl₂ and Na₂IrCl₆H₂O salts through the cell. The measured polarization curves showed smaller cathode overpotential when compared to the first study due to the lower average current density (larger active area) at the cathode. Nevertheless, most losses in all of the de Leon polarization curves were due to the cathode activation overpotential, followed by the ohmic overpotential.

Urian et al. built a DBFC with the cell topology shown in Figure 1.1 with rectangular cross-section flow channels cut into graphite plates and separated by a Nafion 115 membrane [1]. The anode and cathode catalysts were the same Pd:Ir alloy, which was electrodeposited onto the channel walls *in situ* by circulating appropriate salts, as in the work of de Leon. Urian recirculated the fuel and oxidizer

solutions and injected reactants upstream of the cell at rates sufficient to ensure concentrations of 0.1 M at the cell exits. The fuel and oxidizer solutions were both alkaline, making the dominant half-cell R 1.2 and R1.7. The cell was operated at one current density ($25 \text{ mA}\cdot\text{cm}^{-2}$) and the experiments focused on conversion efficiency and long-term performance. Ag/AgCl reference electrodes in contact with the recirculating electrolytes were used to measure the anode and cathode electric potentials.

In [1], substantial water transfer from fuel to oxidizer solution through the membrane was observed. Urian found that increasing the NaOH concentration in the fuel solution from 1 M to 4 M created sufficient electro-osmotic potential across the membrane to reverse the net H_2O flux, so that it flowed from oxidizer to fuel. The increase in NaOH concentration also improved the conversion efficiency by decreasing the rate of H_2 production at the anode, presumably by favoring rates of R 1.2 over R 1.4. Similarly, decreasing the BH_4^- concentration yielded higher fuel conversion efficiency as less BH_4^- was available for R 1.4. With respect to long-term performance, Urian noted a $3 \text{ mV}\cdot\text{hr}^{-1}$ rise in the anode potential vs. Ag/AgCl, which was attributed to oxidized B species blocking anode catalyst sites. The rise in anode potential, while decreasing power output, did improve the fuel conversion efficiency by shifting the relative rates of reaction at the anode in favor of R 1.2.

A second experiment reported by Urian et al. [15] was similar to [1], but with different catalyst layer topology. Both catalyst layers were microfibrous carbon with electrodeposited Pd:Ir, as in the cathode catalyst reported by de Leon [14]. Furthermore, carbon cloth was added between the membrane and catalyst layers to

drive the reactant solutions through the “forests” of upright catalyst-coated carbon structures in the channels. Cell performance without the carbon cloth was comparable to [1], but adding the carbon cloth decreased electrode overpotentials by a factor of 2.1 by forcing the reactant solutions into contact with the entire catalyst surface area.

A summary of relevant prior experiments with aqueous $\text{NaBH}_4/\text{NaOH}$ fuel and H_2O_2 oxidizer is provided in Table 1-1.

Table 1-1. Summary of DBFC experiments with aqueous $\text{NaBH}_4/\text{NaOH}$ fuel and H_2O_2 oxidizer

Source	Anode and Cathode	Membrane	Reactants
Raman [17]	Anode: $\text{MmNi}_{3.55}\text{Al}_{0.3}\text{Mn}_{0.4}\text{Co}_{0.75}^*/\text{C}$ Cathode: Pt/C. Both deposited on carbon paper and hot pressed onto membrane.	Nafion 117	Fuel: 10 wt % NaBH_4 20 wt % NaOH Oxid: 15% w/v H_2O_2 varied H_2SO_4
de Leon	Anode: Au/C on carbon cloth Cathode: Pt/C on carbon paper [16] Pd:Ir on microfibrous carbon [14]	Nafion 117	Fuel: 0.1-2.0 M NaBH_4 6 M NaOH Oxid: 0.05-0.45 M H_2O_2 1 M HCl
Urian [1]	Anode: Pd:Ir electrodeposited on graphite plates Cathode: Pd:Ir electrodeposited on graphite plates	Nafion 115	Fuel: 0.1 M NaBH_4^\dagger 1-8 M NaOH Oxid: 0.1 M $\text{H}_2\text{O}_2^\dagger$ 1 M NaOH
Urian [15]	Anode: Pd:Ir electrodeposited on microfibrous carbon Cathode: Pd:Ir electrodeposited on microfibrous carbon	Nafion 115	Fuel: 0.2 M NaBH_4^\ddagger 4 M NaOH Oxid: 0.2 M $\text{H}_2\text{O}_2^\ddagger$ 4 M NaOH

*Mm: La (30 wt%), Ce (50 wt%), Nd (15 wt%) and Pr (5 wt%).

[†]Target values for fuel cell outlet ports; fuel injected with 2 M NaBH_4^- and 8 M NaOH, oxidizer injected with 50% H_2O_2 .

[‡]Target values for fuel cell outlet ports; fuel injected with 5% NaBH_4 and 1 M NaOH, oxidizer injected with 17% H_2O_2 .

These experiments demonstrate several aspects of DBFC operation that a model should capture in order to make useful performance predictions. Among them are:

- Multiple reactions (or branching pathways) take place at each electrode, leading to electro-oxidation of some reactants and hydrolysis or decomposition of others.
- The rate of H₂ production at the anode decreases with increasing pH. The rate of O₂ production at the cathode decreases with decreasing pH.
- Cell performance is strongly influenced by transport of species to/from the bulk reactant flows, as shown by the planar vs. microfibrous electrode comparison.
- The polarization curves of DBFCs with Nafion membranes are strongly influenced by the membrane resistance when reactant concentrations are high.
- The rate of water crossover through the membrane is large, and is influenced by (at least) electro-osmotic drag and diffusion/permeation.

1.5.2 The Borohydride Oxidation Mechanism on Au

The electro-oxidation of BH₄⁻ is a complex process involving many steps and intermediate species, with ample opportunity for reaction pathways to deviate from R 1.2. Each of the eight electrons is likely transferred to the anode in a single step, suggesting that the mechanism has at least eight steps [36-40]. Some intermediate species, such as BH₃(OH)⁻, are soluble in water and may leave the anode surface prior to complete oxidation [38, 39, 41, 42]. Furthermore, there must be adsorption and desorption reactions delivering reactants to the anode and removing products through

the electrochemical double layer [4]. The rates of adsorption and desorption may depend on the anode electric potential, because the species involved are ions [4, 36]. Finally, there is evidence for reactions among surface adsorbed species such as BH_4^* , OH^* and/or H^* [43, 44]. The complete reaction mechanism and its dominant pathways remain a matter of much debate.

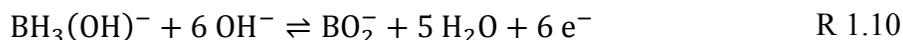
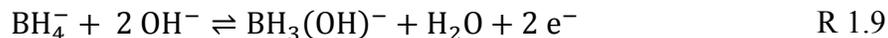
The complexity of the BH_4^- electro-oxidation reaction has several major implications for this study. In particular, the existence of multiple reaction pathways:

- Obscures the rate-determining step (rds), which may change with operating conditions. This uncertainty complicates attempts to predict rates of reaction.
- Leads to varying extents of oxidation that depend on operating conditions. Even if the rate of BH_4^- consumption is known, the relationship between reaction rate and current density may not be straightforward.
- Makes multiple simultaneous charge transfer reactions possible, so that the anode experiences a mixed potential. The anode potential is then a function of the relative rates of multiple reactions, and even the open circuit value cannot be predicted by thermodynamics alone.

Despite these challenges, investigations of the BH_4^- oxidation mechanism have provided some insight into factors controlling the rate, path, and extent of reaction. Moreover, some studies have identified conditions which favor the complete BH_4^- oxidation reaction, making R 1.2 an accurate description of the anode reaction.

In the 1960's, Elder and Hickling [45] and Gardiner and Collat [46, 47] reported polarography (voltammetry) studies which suggested that the electrochemical

oxidation of BH_4^- on Pt involved at least two steps and the intermediate $\text{BH}_3(\text{OH})^-$. The mechanism is given here as R 1.9 and R 1.10.



They also reported values of $n_e < 8$ indicating incomplete oxidation, either due to $\text{BH}_3(\text{OH})^-$ escaping into solution, or competition with hydrolysis (R 1.4).

In 1992, Mirkin et al. [48] used fast-scan cyclic voltammetry, scanning electrochemical microscopy and simulations of adsorption to conclude that BH_4^- oxidation on Au begins with a three step electrochemical-chemical-electrochemical (ECE) process (R 1.11 to R 1.13). In the first step, BH_4^- adsorbs on the anode and loses an electron to become BH_4^* . The second step is a fast chemical hydrolysis with OH^- from solution to form BH_3^- and H_2O . BH_3^- can escape the surface into the bulk solution, or be oxidized to BH_3^* in the third electrochemical step. Presumably the 8 e^- complete oxidation of BH_4^- to BO_2^- involves additional oxidation steps following R 1.13.



Mirkin suggested that adsorption may be rate controlling, pointing out that a surface coverage fraction as small as $\theta_{\text{BH}_4^-} = 10^{-4}$ leads to a linear relationship between $\theta_{\text{BH}_4^-}$ and i , given the measured rate parameters for R 1.11.

More recently, Chatenet et al. reported several studies [38, 41, 42, 49] of mechanisms and rates for BH_4^- oxidation on Au. In [49], voltammographic techniques with a rotating disc electrode (RDE) revealed only one peak for BH_4^- oxidation on Au and $n_e \approx 7$, suggesting little loss to escaping intermediates such as $\text{BH}_3(\text{OH})^-$, which would be rapidly removed by the RDE hydrodynamics. He concluded that all intermediate species remain adsorbed on the Au electrode, under the conditions examined (10^{-2} M NaBH_4 in 1 M NaOH), contradicting the results of Mirkin [48].

In [38], Chatenet fitted rate parameters for an alternative mechanism to electrochemical impedance and RDE cyclic voltammetry measurements. In the new mechanism, the rate of BH_4^- oxidation on Au was determined by electrochemical adsorption of BH_4^- competing for surface sites with the electrochemical adsorption of OH^- . These adsorption reactions should accelerate at high anode potentials as the anions interact with the less-negative electrode, in agreement with experimental evidence. This study suggested potential-dependent adsorption of reactant anions may play an important role in determining the reaction pathway and rate of reaction.

The roles of catalyst layer morphology and transport were explored by Chatenet et al. [42] by RDE voltammetry studies with catalyst layers of varying thickness and porosity. Chatenet shows that the coulombic efficiency of BH_4^- oxidation increased with catalyst layer thickness and porosity, which was attributed to large residence

times for intermediate species. The longer residence times raised the rates of subsequent adsorption and oxidation, increasing the net number of electrons provided by each BH_4^- anion.

Krishnan [50] used rotating ring-disc electrode (RRDE) voltammetry to study intermediate species produced in the oxidation of BH_4^- on Au. With BH_4^- oxidized at the Au disc, intermediate species soluble in water were swept past the ring by the RRDE hydrodynamics, where further oxidation steps yielded additional current. The onset of non-zero current density at the ring coincided with the onset of BH_4^- oxidation at the disc, but the ring potential was negative to the disc potential, indicating the oxidation of intermediate species with more negative E^0 . The ring potential range showing the greatest current suggested the predominant intermediate species was $\text{BH}_3(\text{OH})^-$. Spread in the ring electrode current density peak indicated the presence (in lower concentration) of other oxidation intermediates with different E^0 , suggesting desorption of other intermediates from the Au disk on the path to BO_2^- .

Concha et al. [43, 51] employed Fourier transform infrared spectroscopy (FTIR) to identify intermediate species adsorbed on the Au surface during BH_4^- oxidation, and measure the relative abundance of each species as a function of electrode potential. In a sweep from -200 to 1400 mV vs. RHE in 1 M NaBH_4^- / 1 M NaOH, Concha found sequential majority species on the surface in the order expected for BH_4^- oxidation: BH_3 , BH_2 , ... BO_2^- . While BH_4^- hydrolysis took place and produced BH_3 on the surface at potentials below 200 mV vs. RHE, the rate was slow. No current was detected below 200 mV vs. RHE. In the potential range 200 to 500

mV vs. RHE, the abundances of BH_3 and BH_2 rose dramatically, coinciding with the appearance of current due to electro-oxidation of B species and increasing hydrolysis rate. Above 500 mV vs. RHE, greater amounts of BH_3 on the surface and the appearance of B-O bonds suggested to the authors that the complete BH_4^- oxidation reaction dominated in this potential range. Concha suggests that the onset of BH_4^- reactions occurs when the anode potential becomes high enough for BH_4^- to overcome the electrostatic repulsion of the double layer (activation energy barrier for adsorption) to reach the anode. Once BH_4^- reaches the anode, it proceeds through a series of oxidation steps. As the anode potential is raised, the oxidation steps favor charge transfer to the anode rather than hydrolysis. This may be due, in part, to the anionic intermediate species having lesser propensity to desorb at less negative anode potentials.

The last study of BH_4^- oxidation on Au to be discussed here was reported by Rostamikia et al. [36, 37, 52]. Rostamikia used density functional theory (DFT) to estimate the free energies of aqueous and surface-adsorbed species involved in BH_4^- oxidation. The relative energies were then used to identify thermodynamically favorable paths from aqueous BH_4^- to final oxidation products. Several of Rostamikia's conclusions are relevant to this study. First, BH_4^- oxidation on Au begins with adsorption of the aqueous species at a rate which depends on the anode potential [37] as in R 1.11. Adsorption is followed by breaking B-H bonds, yielding surface adsorbed BH_x^* species ($0 \leq x \leq 4$) and H^* . The BH_x^* species either lose additional H to form more H^* or get hydroxylated by OH^- (aq) to form species such as $\text{BH}(\text{OH})_2^*$. The ultimate product is $\text{B}(\text{OH})_4^-$, the hydrated form of BO_2^- . The

mechanism proposed by Rostamikia consists of elementary (one e^-) charge transfer, dehydrogenation and hydroxylation steps, yet DBFC anode reactions are often cast in terms of global (multi-electron) R 1.2 and R 1.4. The relative rates of global R 1.2 and R 1.4 can be understood in the context of the mechanism proposed by Rostamikia, which indicates they depend on the fate of H^* . The surface adsorbed hydrogen can undergo one of two reactions to leave the surface [52]:



or,



Several observations can be made regarding R 1.14 and R 1.15:

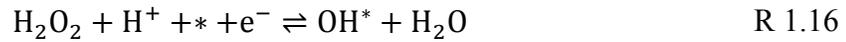
- R 1.14 is second order in H^* whereas R 1.15 is first order, so higher BH_4^- concentration (which raises the BH_4^- adsorption rate and yields more H^*) should favor R 1.14 and H_2 production.
- Higher OH^- (higher solution pH) should favor R 1.15.
- Higher (less negative) anode electric potentials should favor electrochemical oxidation (R 1.15).

All of these observations have been noted by published experimental studies, which lend credence to the Rostamikia mechanism. This has implications for the way the anode reaction on Au is modeled, because the rate(s) of reaction depend on surface fractions θ_{BH_4} and θ_H .

While BH_4^- oxidation on Au yields H_2 at all potentials in 1 M NaBH_4 / 1 M NaOH [41], complete BH_4^- oxidation (reaction R 1.2) may be accurate under certain combinations of electrocatalyst and operating conditions. Gardiner and Collat [47] suggested that the oxidation of BH_4^- on Au favors reaction R 1.2 in strongly alkaline conditions and that reaction R 1.4 may be insignificant at $\text{pH} \approx 14$. Both Cheng and Scott [53] and Liu et al. [54] reported that an important metric for the relative rates of R 1.2 and R 1.4 is the ratio $C_{\text{OH}^-}/C_{\text{BH}_4^-}$; when $C_{\text{OH}^-}/C_{\text{BH}_4^-} > 5$, the complete borohydride oxidation dominates. This suggests that sufficiently high OH^- concentrations can strongly bias the mix of anode reactions in favor of reaction R 1.2.

1.5.3 The Hydrogen Peroxide Reduction Mechanism on Pd:Ir

The borohydride oxidation mechanism has received greater attention in the DBFC literature, but the mechanism for H_2O_2 reduction also plays an important role. Two pathways for H_2O_2 reduction on precious metal catalysts have been observed [55, 56]; the first “normal” pathway consists of R 1.16 and R 1.17, which together consume two electrons from the cathode.



In the second, autocatalytic pathway, R 1.17 is replaced by R 1.18. Both pathways involve adsorbed OH (OH^*), but the second pathway differs in that H_2O_2 reduction makes more OH^* available to reduce additional H_2O_2 .



The autocatalytic pathway has been observed on Ag, but not on Pt or Pd [57]. No papers examining the mechanism of H₂O₂ reduction on Pd:Ir alloys could be found, however other properties of these alloys suggest that the mechanism for H₂O₂ reduction on Pd:Ir also follows the normal pathway. Adding Ir to the Pd cathode catalyst has been shown to decrease the rate of H₂O₂ decomposition, with a concomitant decrease in activity for H₂O₂ reduction [14, 58]. Greater activity would have been expected if Ir induced an autocatalytic effect which Pd does not normally exhibit.

1.5.4 Additional DBFC Electrode Reactions

R 1.2 (BH₄⁻ electro-oxidation) and R 1.4 (BH₄⁻ hydrolysis) dominate at the anode. R 1.3 (H₂O₂ electro-reduction) and R 1.5 (H₂O₂ decomposition) dominate at the cathode. These are not the only reactions which can occur, however. The electrode potentials can reach values which drive reactions involving the solvent (H₂O) or supporting electrolytes (NaOH or H₂SO₄). The simplest way to evaluate the possibility of such reactions is often to examine the relevant pH-*E* (Pourbaix) diagram.

The anode potential at open circuit should be near the standard equilibrium potential for BH₄⁻ electro-oxidation ($E_{R1-2}^0 = -1.24 \text{ V vs. RHE}$) if it is the dominant charge transfer reaction. Referring to the Pourbaix diagram for H₂O (Figure 1.7), an electrode having this potential in a strongly alkaline medium should drive R 1.20, which at pH 14 and $a_{\text{H}_2} = 1$ has $E_{R1.20}^0 = -0.828 \text{ V vs. RHE}$. The result is a “mixed

potential” where the equilibrium potential of the anode depends on the relative rates of R 1.2 and R 1.20 and falls in the range $E_{R1-2}^0 < E^0 < E_{R1.20}^0$. Ordinarily R 1.20 is written as a dynamic equilibrium with rates in each direction, however Au has little propensity for breaking H-H bonds, so the oxidation of H₂ (reverse of R 1.20) is unlikely.

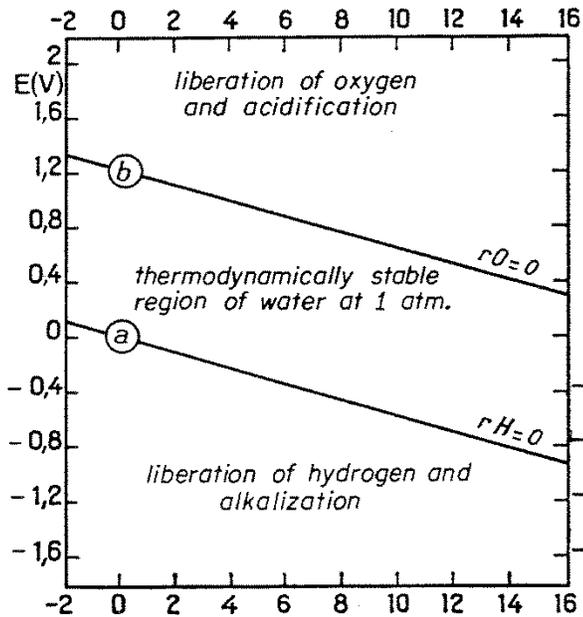


Figure 1.7. Pourbaix diagram showing regions of stability, oxidation and reduction of H₂O as functions of pH and electrode potential [59].



A similar scenario exists at the cathode, where the standard reduction potential of H₂O₂ in a strongly acidic medium is $E_{R1-3}^0 = 1.77 \text{ V vs. RHE}$. Figure 1.7 shows that R 1.21 should take place at electrode potentials above $E_{R1.21}^0 = 1.23 \text{ V vs. RHE}$ for pH = 0 and $a_{\text{O}_2} = 1$.



From a mechanistic point of view however, R 1.21 is properly modeled as the summary of R 1.22 and R 1.3, with H_2O_2 being an intermediate in the O_2 reduction process at potentials more negative than $E_{R1.21}^0$. This gives the intermediate H_2O_2 an opportunity for other interactions prior to the second step, which is more realistic.



Additionally H^+ reduction (R 1.23) can occur at the cathode. The use of low pH oxidizer solutions provides abundant H^+ in solution, which can be reduced to H_2 at potentials below $E_{R1.23}^0$ (see line *a* in Figure 1.7).



1.5.5 Rates for BH_4^- oxidation on Au and H_2O_2 reduction on Pd:Ir

At least three efforts to measure the kinetic rate parameters for BH_4^- electro-oxidation have been reported, however the complexity of this reaction introduces substantial uncertainty in the results. Chatenet [49] used linear voltammetry and chronometric techniques with an RDE to measure i_0 and n_e in 0.1 to 1.0 M NaOH and 10^{-2} to 1 M NaBH_4 at 25°C . The result was $i_0 = 7.4 \times 10^{-6} \text{ A}\cdot\text{cm}^{-2}$ and $n_e = 7$ for 10^{-2} M NaBH_4 in 1 M NaOH.

Santos [60] applied chronocoulometric techniques to measuring the exchange current density i_0 , symmetry factor β_a and standard rate constant k_a at an Au disk in

2 M NaOH with $C_{BH_4} = 0.03, 0.06, 0.09, 0.12$ M. The result at 25°C was values of β_a ranging from 0.07 to 0.13 and values of i_0 ranging from 16.4 mA·cm⁻² to 0.46 mA·cm⁻². Santos obtained values for k_a by noting that the anodic current at equilibrium is equal to i_0 and the number of electrons transferred in the rds is likely 1:

$$i_0 = k_a n_e F C_{BH_4} \exp(\beta_a n_e f E_{R1.2}^0) \quad \text{Eq. 1.24}$$

In a second paper [61] describing the same experiments, Santos concluded that in the examined range of concentrations at 25°C, the BH₄⁻ electro-oxidation reaction was irreversible, diffusion controlled and the rds involved the transfer of one electron.

Finkelstien et al. studied the rates of both BH₄⁻ oxidation at Au [39] and H₂O₂ reduction at Pt [62]. For BH₄⁻ oxidation, Finkelstien reported $k_a = 6.4 \times 10^{-4}$ m·s⁻¹ and $\beta_a = 0.22$ at -0.230 V vs. RHE in 5 mM NaBH₄ / 1 M NaOH. For H₂O₂ reduction, Finkelstien reported $k_c = 8 \times 10^{-3}$ m·s⁻¹ and $\beta_a = 0.45$ at 0.6 V vs. RHE in 5 mM H₂O₂ / 0.5 M H₂SO₄. The authors did note, however, that the value of k_c was unphysically high and did not provide a rationale as to why.

Cao [21] studied the kinetics of H₂O₂ reduction at low pH on Pd nanoparticles immobilized on an Au disk. An exchange current density of 3.8×10^{-4} mA·cm⁻² was reported for 30 mM BH₄⁻ in 0.1 M H₂SO₄ at a temperature of 293 K.

1.5.6 Transport through Nafion Membranes in DBFCs

Most reported DBFC experiments have separated the fuel and oxidizer with a Nafion cation exchange membrane [20, 25]. Nafion membranes are extruded sheets of polytetrafluoroethylene (PTFE) and polysulfonfyl fluoride vinyl ether copolymer; the PTFE is a backbone for SO₃⁻ functional groups which provide the cation exchange

properties. In the picture expounded by Newman and Weber [63], the cationic conductivity of Nafion depends on the presence of pores containing water. The pores are effectively lined by SO_3^- groups in the membrane matrix, which balance the charge of hydrated cations in the pores. Anions are excluded from the pores by the local negative charge of the pore walls, which establishes a Donnan potential at the membrane-solution interface. Important Nafion transport properties include the average number of H_2O molecules per SO_3^- group (λ) and the number of H_2O molecules in the hydration shell of each cation (n_d). Increases in λ imply wider or a larger number of pores, and correlates with ionic conductivity [64]. The value of n_d depends on the cationic species and is known as the “electro-osmotic drag coefficient” because it describes the number of water molecules transported through the membrane by each cation [65].

Transport of cations through Nafion membranes occurs due to diffusion and migration, while transport of water occurs due to diffusion and electro-osmotic drag. All species are subject to permeation, where a pressure differential across the membrane drives a bulk flow through the pores, carrying some or all constituents of the higher pressure fluid. Some transport of anions in response to concentration or pressure gradients is possible with anion concentrations lower than $C_{\text{SO}_3^-}$ in the membrane because the Donnan potential at the membrane-solution interface acts as an activation energy barrier to entry. The Donnan potential can be overwhelmed when the anion concentration exceeds the concentration of SO_3^- groups in the Nafion, which then admits anions to the pores.

The conductivity of the membrane is related to the mobilities of the charge carriers (cations) flowing through it, as shown for aqueous electrolytes in §1.2.2. The ohmic losses observed in DBFC experiments stem in large part from the mobility of Na^+ in Nafion, which is lower than the mobility of H^+ due to its larger hydration shell ($n_{d,\text{Na}^+} > n_{d,\text{H}^+}$).

In an operating DBFC, the electric potential gradient in the membrane is oriented such that the migration flux of Na^+ ions flows from fuel solution to oxidizer solution (see Figure 1.8). This electric potential gradient acts to keep fuel solution anions on the fuel side of the membrane and cations in the oxidizer solution on the oxidizer side of the membrane. Studies of BH_4^- crossover in DBFCs using Nafion 117 membranes have shown that the rate of crossover diminishes with increasing current density (electric potential gradient in the membrane) [66].

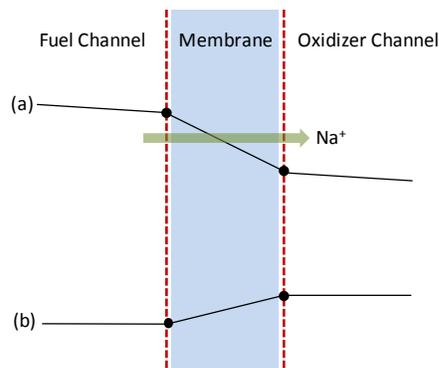


Figure 1.8. Illustration of the electric potential profile through the membrane when the DBFC is operating (a) and a possible profile at open circuit (b), which would encourage crossover of other species.

When the cell is at open circuit, however, the rates of crossover for species other than Na^+ may rise. The electric potential gradient across the membrane and rates of transport through the membrane at open circuit are dictated by the electrochemical

potentials of the species on each side. Large differences in concentration can provide strong driving forces for crossover, and if $\Delta\tilde{\mu}_{k,mem} < 0$, then species k will cross the membrane. It is possible for steady state at open circuit to include fluxes of charged species through the membrane even if the net current into/out of the electrodes is zero. Experiments have suggested that crossover of species such as BH_4^- and OH^- may contribute to open circuit values of lower than $E_{\text{R1.1}}^0$ by creating a mixed potential at the cathode.

Studies [64, 67] of Nafion membranes in contact with aqueous electrolyte solutions containing Na^+ and H^+ have shown that the cation electronic mobilities and electro-osmotic drag coefficient depend on $X_{\text{H}^+,mem}$ and $X_{\text{Na}^+,mem}$. The electronic mobilities in Nafion 115 have been described [67] in terms of their mole fractions and an interaction parameter k_{int} :

$$u_{\text{Na}^+,mem} = u_{\text{Na}^+,mem}^0 (1 - k_{int}X_{\text{H}^+,mem})/F \quad \text{Eq. 1.25}$$

$$u_{\text{H}^+,mem} = u_{\text{H}^+,mem}^0 (1 - k_{int}X_{\text{Na}^+,mem})/F \quad \text{Eq. 1.26}$$

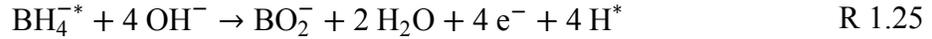
Faraday's constant is included in Eq. 1.25 and Eq. 1.26 to convert from electronic mobility to standard ion mobility. The electronic mobilities with only one type of cation in the membrane were found in [67] to have the following values: $u_{\text{Na}^+,mem}^0 = (2.7 \pm 0.1) \times 10^{-8} \text{ m}^2 \cdot \text{V}^{-1} \cdot \text{s}^{-1}$ and $u_{\text{H}^+,mem}^0 = (1.49 \pm 0.03) \times 10^{-7} \text{ m}^2 \cdot \text{V}^{-1} \cdot \text{s}^{-1}$. The interaction parameter was found to be $k_{int} = 0.20 \pm 0.02$.

1.5.7 Mathematical Models of DBFCs and Similar Fuel Cells

DBFC models have been published prior to this work. Verma and Basu [68] modeled an O_2 DBFC consisting of a Nafion membrane in contact with porous

catalyst layers and a well mixed volume of fuel. Mass transport to the cathode was governed by Fick's law (O_2 diffusion in air) and transport to the anode was neglected. All charge transport processes in the cell were described by a fit to measurements of ohmic resistance. Sanli et al. [69] published a similar model, but with H_2O_2 as the oxidizer and mass transport neglected entirely. Both [68] and [69] ignore down-the-channel effects. Shah et al. [70] published a DBFC model with the most detailed treatment of the electrode reactions to date, for a cell topology similar to that of PEMFCs (as shown in Figure 1.6, right side). The cell consisted of a Nafion membrane with a porous Pt catalyst layer on each side; the catalyst layers were separated from reactant flow channels by porous diffusion layers. The model predicts cell voltage at a given current density by calculating activation and ohmic overpotentials explicitly as functions of current density, and then subtracting the overpotentials from the open circuit voltage. Concentration overpotentials were included by estimating concentrations near the electrodes. Species concentrations were assumed to be uniform in the flow channels, but transport from the flow channels to the catalyst layer was approximated by a Nernst diffusion layer thickness correlation. Down-the-channel effects were not addressed. The anode reaction model included BH_4^- adsorption (R 1.24), partial electro-oxidation of BH_4^- to yield $4 e^-$ and H_2 (R 1.25) and the complete Tafel-Volmer-Heyrovsky mechanism for H_2 evolution (R 1.26 to R 1.28) . Mixed potentials at the anode were handled by allowing all of the reaction rates to find equilibrium at a common anode potential for the specified current density. Shah found that $\theta_{BH_4^-}$ tended to be quite small ($\sim 10^{-4}$) due to slow adsorption of anions on the negative anode and rapid dehydrogenation of

the adsorbed BH_4^- . These modeling results concur with the experimental results of Mirkin et al. [48] and Chatenet et al. [44].



Byrd and Miley [71] reported a 2D finite element DBFC model used for parametric design analysis with respect to the cell geometry. The modeled DBFC consisted of a Nafion membrane coated with catalyst on each side (see Figure 1.6, right), with porous electrically conductive diffusion layers in contact with each catalyst layer. The model domain included the membrane, catalyst layer and diffusion layers between two channels; i.e. under the “land” of the flow field. The goal was to model transport through the porous diffusion media between the channels to evaluate the effects of parameters such as channel spacing and diffusion medium porosity. The Byrd study excluded transport in the channels and migration by treating electrolytes (for example, NaOH) as single uncharged species in aqueous solution. Transport in the diffusion media was described by a combination of Fick’s Law for diffusion and Darcy’s law for creeping flow through a porous medium. Convection and the interactions of ions in solution were ignored. The membrane was treated as an ohmic resistance with H^+ flowing from anode to cathode. The reason for charge balancing the cell with H^+ was unclear, as Na^+ is widely accepted as the

charge carrier through Nafion membranes in DBFCs. Judging by the global reactions cited in the paper, both reactant solutions were treated as if there were no supporting electrolyte and $\text{pH} = 7$, which is an unrealistic scenario given the rapid rates of BH_4^- hydrolysis and H_2O_2 decomposition under such conditions. No comparison to experiments was provided.

Of prior DBFC models, only Shah [70] and Byrd [71] included transport in a meaningful way, yet neither addressed down-the-channel effects on cell performance or the complex interplay of convection, diffusion and migration for the ionic species involved. The reaction rate model described by Shah appears to show good agreement with experimental results for a Pt anode, but is inappropriate for the Au anode in this study. No prior DBFC model has studied the geometry depicted in Figure 1.

Other fuel cell topologies and chemistries have been studied using 2-D finite volume models. A relevant example is that of Sprague and Dutta [72, 73], who modeled a fuel cell similar to Figure 1.1 with the exception that it had no membrane. The cell had a single liquid electrolyte and electrically neutral fuel and oxidizer species, which reacted in a generalized way governed by the Frumkin-Butler-Volmer equation. The Poisson-Nernst-Planck equations were solved to resolve electrochemical double layers near the electrodes and the distributions of species and charge in the channel. The model presented here is solved in a similar way, but it differs from that of Sprague and Dutta in several key aspects. The model in this study includes a membrane, has different electrolytes and chemistries (specific to a DBFC)

at the anode and cathode, and solves for the electric field by explicitly enforcing electroneutrality rather than by solving Poisson's electrostatic equation.

1.6 The Present Study – A Step toward Understanding DBFC Performance

DBFC performance can be characterized by peak power density, thermodynamic (voltage) efficiency and reactant utilization. Despite progress over the past decade, all three performance characteristics remain inadequate for practical applications. A key challenge to further progress is poor understanding of the processes that dictate DBFC performance and how these processes change with operating conditions and cell geometry. Prior studies have yielded some clues, yet these studies, while helpful for identifying important sources of losses, do not relate them to cell-level performance and design. Knowledge of these relationships will point to specific ways to improve cell design and operation for improved performance.

Peak power density is a good metric illustrating the gap in the present understanding of DBFC performance. Peak power density depends on the rates of diffusion, migration and advection in the cell channels. In a DBFC, these transport processes are not well understood because they are difficult to measure independently. Furthermore, competing side reactions can change the fate of reactants upon arrival at the electrodes, so that current density is often less than that predicted by transport alone.

Experimental and modeling studies leads to deeper understanding when model predictions are corroborated with experimental results. This is an iterative process that continues until the mental picture embodied by the model converges toward an

accurate description of the real process. The resulting calibrated model can then be used for investigating relationships among variables, which are difficult to access in an experiment. For example, the diffusion, migration and advection fluxes can be calculated separately and then compared to identify the dominant transport process.

This study attempted to do that by not only comparing DBFC model results to new experiments (following the topology of Figure 1.1) reported herein but also to relevant studies with similar electrodes discussed in the review above. The initial DBFC cell-level modeling attempted to predict cell performance with ideal reactions and did not match the experimental studies well, but provided a basis for understanding how other processes could impact and limit cell performance. From there, more detailed reaction modeling was incorporated in to the cell-level DBFC model in order to elucidate how competitive reactions at both the anode and the cathode coupled with transport mechanisms to impact cell voltage and fuel utilization.

The top-level goals of this study are listed here:

- (a) *Develop tools for investigating the processes governing DBFC performance.*
- (b) *Identify DBFC performance trends, which depend on transport in the channels and/or membrane.*
- (c) *Evaluate the limits of DBFC performance, assuming present efforts to develop more active and selective catalysts and higher conductivity membranes are successful.*
- (d) *Relate the impact of competing side reactions to cell design and operating parameters, so that their influence can be mitigated.*

These goals were approached by a series of related tasks. First, a transport-focused model of the DBFC illustrated in Figure 1.1 was developed (Chapter 2), and then used to analyze a DBFC with ideal electrode reactions (Chapter 3). A single-cell DBFC having the same geometry as the model was constructed and then used for experimental studies examining the influences of transport and competing reactions on performance (Chapter 4). The DBFC model was calibrated to the single-cell experiments, and then used to investigate the influence of competing reactions on DBFC performance in ways that would have been difficult through experiments alone (Chapter 5).

There were several valuable products of this study, including the calibrated model and experimental data. The most significant contributions were a better understanding of the factors dictating DBFC performance and specific strategies for improving it. These can bring DBFCs closer to practicality by guiding design of higher performance prototypes and future research efforts.

Chapter 2: Mathematical Model of a DBFC

A steady-state DBFC model with sufficient detail to capture the transport processes most affecting performance was designed here to have sufficient flexibility in the descriptions of electrode reactions to enable evaluation of different approaches to modeling those reactions. This chapter describes the model and solution approach, and provides the relevant thermodynamic, kinetic and transport parameters.

2.1 *Model Summary*

The steady state performance of a DBFC having the form shown in Figure 1.1 was modeled with a 2-D finite volume approach. The model domain includes two rectangular channels separated by a membrane and channel walls which function as electrodes (see Figure 2.1). The catalyst layers are homogeneous, isotropic, non-porous planes with a surface roughness characterized by the ratio of geometric to electrochemical surface area ℓ . Fuel and oxidizer solutions flow through the channels in the same direction. Reactions occur at the channel walls while Na^+ and H_2O pass through a Nafion membrane from fuel channel to oxidizer channel.

2.1.1 Simplifications and Assumptions

The operating conditions and cell geometry in this study enable several simplifications. The fuel and oxidizer solutions are liquids and therefore effectively incompressible. Reactant channel flows are laminar because the Reynolds numbers are less than 150 for flow rates reported in the literature and in the experiments of this

study (discussed in Chapter 4). The fuel and oxidizer solutions are electrically neutral because the electrochemical double layers are thin and treated as part of the electrode interfaces. The electrodes are good electrical conductors and therefore all points along the channel for each electrode share the same electric potential. The side walls of the channels are electrical insulators and inert (support no reactions). Homogeneous reactions in the fuel and oxidizer solutions occur so slowly in high pH ($\text{pH} > 13$) fuel and low pH ($\text{pH} < 1$) oxidizer that they can be neglected without significantly impacting the results [3]. The Nafion membrane is in the Na^+ form and thin enough to make x -direction fluxes in the membrane insignificant.

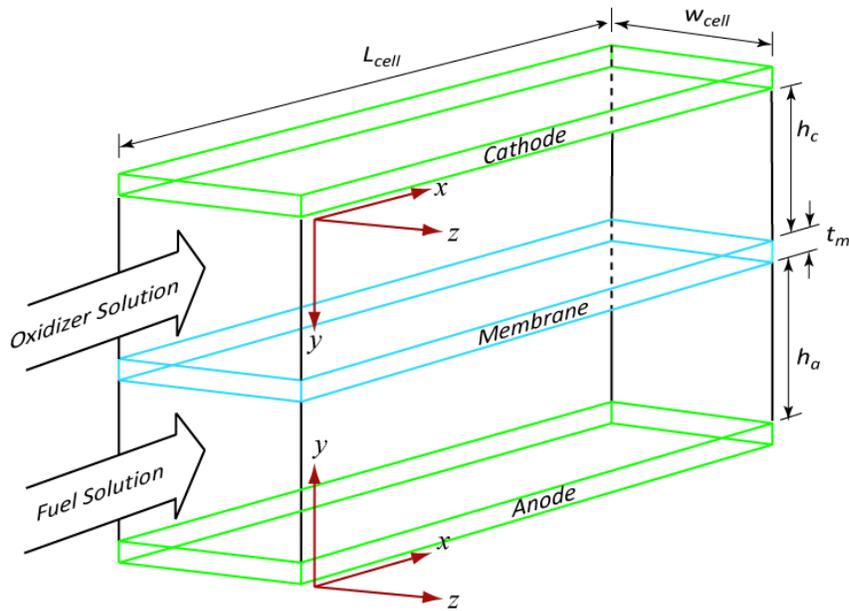


Figure 2.1. Cell geometry and model domain with dimensions and spatial coordinates.

Three simplifying assumptions accommodate unknown transport parameters:

1. The fuel and oxidizer solutions are ideal electrolytes, i.e. $\gamma_k = 1$
2. Only Na^+ and H_2O pass through the membrane (negligible BH_4^- crossover).
3. Heat is removed quickly enough to maintain isothermal conditions.

The influences of activity on transport (Eq. 1.24) and electrode reaction rate (Eq. 1.15) were discussed in Chapter 1. There are several approaches to estimating the effect of other ions on a_k . The most common is the Debye-Huckel law, which models the influence of other ions as a uniform “cloud” of charge surrounding k . The effective local charge density is given by the ionic strength, defined in Eq. 2.1.

$$I = \frac{1}{2} \sum_k z_k^2 C_k \quad \text{Eq. 2.1}$$

The extended Debye-Huckel equation gives γ_k in terms of I and two parameters A and B which describe properties of the solvent; for water at 25°C, $A = 0.5092 \text{ kg}^{0.5} \text{ mol}^{0.5}$ and $B = 3.283 \times 10^9 \text{ kg}^{0.5} \text{ mol}^{0.5} \text{ m}^{-1}$ [74]. A third parameter \bar{a}_k is the effective ionic diameter of species k in water.

$$\ln(\gamma_k) = \frac{-Az_k^2\sqrt{I}}{1 + \bar{a}_k B\sqrt{I}} \quad \text{Eq. 2.2}$$

Unfortunately, the standard Debye-Huckel law (numerator only in Eq. 2.1) is appropriate for solutions having $I < \sim 0.5 \text{ M}$, which is not the case in most DBFC experiments. Above $\sim 0.5 \text{ M}$, the extended Debye-Huckel law gives better agreement with experiment, but at concentrations of several molar the predicted values of γ_k can be in error by more than the assumption $\gamma_k = 1$.

The other standard approach is to employ the Pitzer equations, which account for individual ion pair interactions to give a more accurate description over a wider range of concentrations. The Pitzer equations would be appropriate for the range of concentrations typically found in a DBFC, yet the data for some species (specifically,

BH₄⁻) were not available in the literature. Neither approach was adopted for this work.

Aside from inaccuracy in the Debye-Huckel law and lack of parameters for the Pitzer equations, the ideal solution simplification was employed because the deviations from ideal behavior were not expected to substantially change the results of transport or reaction rate estimates based on Eq. 1.24 and Eq. 1.15. This simplification is appropriate for both the transport and reaction rate calculations when the solutions are dilute, in which case solutes only interact weakly and $\gamma_k \approx 1$. It is still appropriate for at higher concentrations insofar as $\partial \ln \gamma_k / \partial \ln C_k \approx 0$ and any offsets introduced by $\gamma_k \neq 1$ are absorbed into other parameters (D_k for transport and k for reaction rates) in the equations. The activity of H₂O₂ is a good example; $\gamma_{H_2O_2}$ is plotted with respect to X_k in Figure 2.2, which shows that $\gamma_{H_2O_2}$ changes by ~3% over the range 0 to 40 mM, and $\gamma_{H_2O_2}$ is proportional to $C_{H_2O_2}$ over this range. As a result, $\partial \ln \gamma_k / \partial \ln C_k$ is negligible in this concentration range.

The impermeable membrane simplification is reasonable because of the electric field in the membrane at cell potentials other than OCV, as discussed in Chapter 1. The isothermal system assumption is reasonable given that the fluids are predominantly water, which with its high specific heat ($c_{p,H_2O} = 4.186 \text{ kJ kg}^{-1} \text{ K}^{-1}$) can absorb substantial heat without a significant rise in temperature, and given that the rates of heat production are small. With a typical power density of 100 mW cm^{-2} , overall efficiency of 25%, 2.5 cm^2 electrode and 10 mL min^{-1} flow rate, the reactant stream temperature rise from inlet to outlet would be $\sim 0.36^\circ\text{C}$.

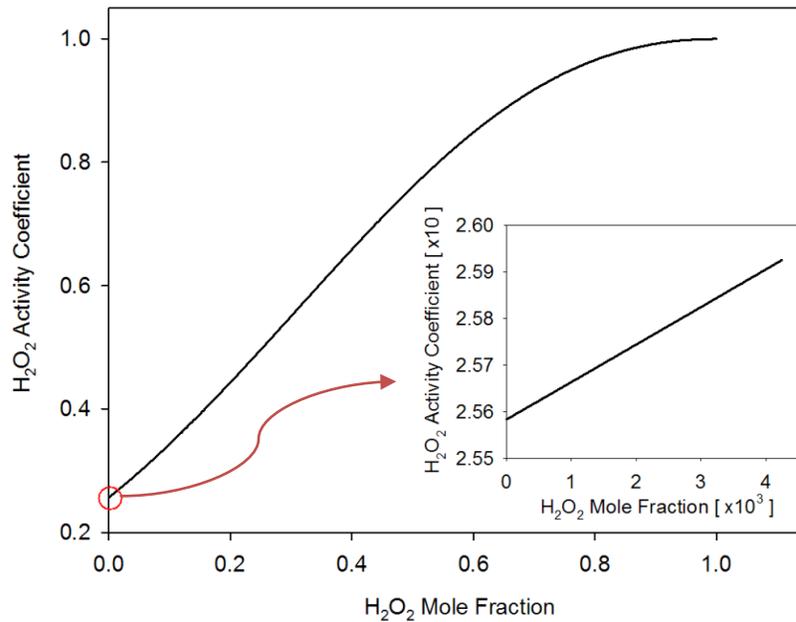


Figure 2.2. Activity of H_2O_2 in water, predicted by empirical fit from [75].

2.1.2 Geometry and State Variables

The model domain is subdivided into channels and interfaces. Channels are discretized into numerical cells with state variables for pressure, x - and y -direction mass-averaged velocities, electric potential, and species mass fraction, as shown in Figure 2.3. Figure 2.1 shows orientations of the x - and y -coordinates. Velocity storage locations in the channels are staggered from scalar variables by one-half discretization to avoid odd-even decoupling, as illustrated by the grid stencil in Figure 2.4. All interfaces are broken into discrete line segments (numerical cells) having state variables for electric potential and species mass fractions. The membrane interfaces also store pressure.

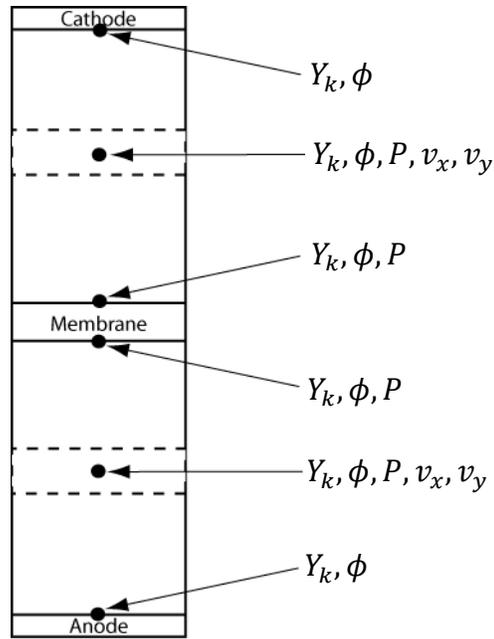


Figure 2.3. Model domain in one x -direction slice across the cell, showing subdomains for channel cells, interfaces and electrodes. Only one channel cell is shown on each side for clarity.

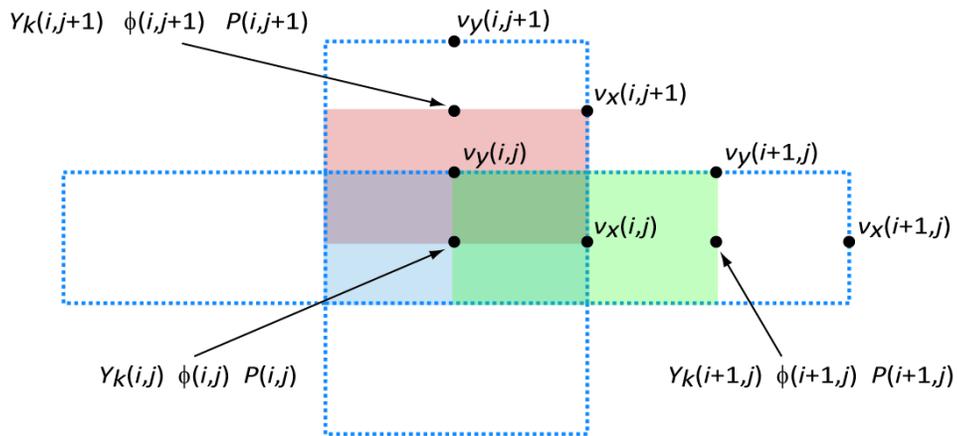


Figure 2.4. Grid stencil showing relative locations of state variables. Cells for Y_k, ϕ and P are marked with dashed lines and one cell is shaded blue. One cell for v_x is shaded green and one cell for v_y is shaded red.

A custom grid generation function was developed to produce numerical grids with uniform, linearly varying or logarithmically varying density depending on user-

set flags. In each case the channel dimensions and the ratio of smallest to mean cell size ($x_{min}/\overline{\Delta x}$ and $y_{min}/\overline{\Delta y}$) are specified, and then the grid function outputs the dimensions and variable locations for all cells in the channel. For the results presented in Chapters 3 and 5, the numerical grid density varied linearly in the x - and y -directions with the highest density near the inlet, electrodes and membranes to capture entrance effects and the development of boundary layers. One such grid is shown in Figure 2.5, which was created by a custom grid visualization function.

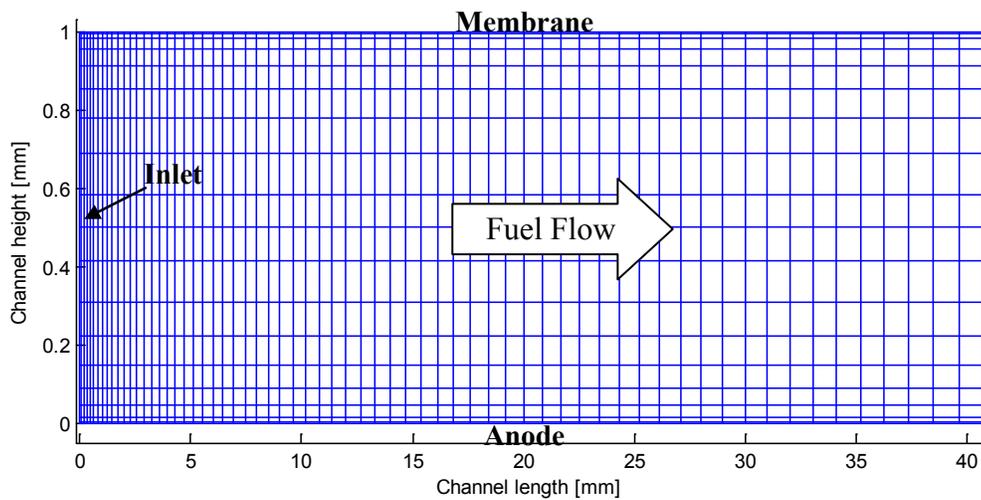


Figure 2.5. Typical channel discretization scheme near the fuel inlet (left).

For most simulations the smallest x -discretization was $0.25\overline{\Delta x}$ and the smallest y -discretization was $0.10\overline{\Delta y}$. The mean aspect ratio ($\overline{\Delta x}/\overline{\Delta y}$) was 68 and the largest was 300. Aspect ratios were smallest near the center of the channel and largest near the electrodes and membrane. Large aspect ratios near the electrodes and membrane did not cause numerical instabilities, presumably because transport in those regions is nearly all in the y -direction. The grid density was increased until the solution achieved grid independence, judged as $<1\%$ change in the predicted current density

with further refinement. Results presented in Chapter 3 with the minimal number of species for global reactions required 49,800 state variables to simulate a 200 mm long channel broken into 150 x -discretizations and 17 y -discretizations.

2.1.3 Solution Approach

A solution is the set of state variable values for each cell which conserve mass, momentum, species and charge and are consistent with the boundary conditions. For a specified cell voltage at the boundary, the state variable solution provides information to calculate the total cell current for a given geometry and operating conditions. The model is set up in MATLAB and solved using a Newton solver in KINSOL. KINSOL is a non-linear equation solver available from the Lawrence Livermore National Laboratory as part of a collection of equation solving tools called SUNDIALS [37]. The MATLAB code sets up the model geometry, discretizes the domain, applies boundary conditions, and provides a function KINSOL calls to evaluate the residuals (errors) associated with each prospective solution. KINSOL runs until the residual 2-norm is driven below a specified threshold (typically 10^{-3}), at which point the solution has been found. KINSOL parameters used with the DBFC model are provided in Table 2-1.

Solution time for a single cell voltage using the native MATLAB code and one 3.3 GHz Xeon processor core is typically ~80 min, but it is reduced to ~40 min when the initial guess is from a solution for similar conditions. Initial guesses were found to have a large impact on the likelihood of finding a solution and the time (number of iterations) required to find it. The electric potential had the greatest affect, with initial guesses far from the solution often causing the solver to fail. The sensitivity to

electric potential in the initial guess limited the size of cell voltage steps from one solution to the next when computing a polarization curve, because each solution was used as the initial guess for the next point. A function was created to read a solution at one cell voltage and adjust the electric potential profiles in both the x - and y -directions to be more similar to those for the next cell voltage in the polarization curve. This function permitted voltage steps to be made four times larger, decreasing the time to compute a polarization curve by a factor of four.

A further improvement in speed was obtained by compiling the most computationally expensive functions into MATLAB exchange (MEX) functions. MEX functions are C-code executables called from within the MATLAB environment, but executed externally. The model with MEX files solves one cell voltage in ~ 1.2 min when starting from a solution for similar conditions. A 10 point polarization curve then requires ~ 12 min, if the initial guess for the solution at each cell voltage is the solution from the previous cell voltage. This is a speed improvement over the native MATLAB code by a factor of 67.

Finally, the structure of the vector \vec{S} containing all of the system state variables is important to the solution approach. The Newton solver can find a solution given an \vec{S} having any order, but the solution process will be substantially faster if the state variables are ordered as they appear in the DBFC. Specifically, the variables should begin in the corner where one electrode and the inlet meet, and then “walk” across the cell to the other electrode. The next set of state variables begins back at the first electrode, and the process repeats, rastering across the entire DBFC. This pattern results from the relationships among adjacent cells. From the perspective of

calculating a Jacobian matrix J of the system's governing equations, the state variables in a computational cell only depend on the values in neighboring cells. Ordering \vec{S} in this way groups the non-zero elements of J into three bands, resulting in a tri-diagonal matrix. Calculating J is the most computationally expensive part of a Newton search algorithm, and if the structure of the tri-diagonal J is known, then many zero elements can be ignored, substantially reducing the computational burden of calculating J and decreasing the solution time.

Solution speed was of interest because the code must be called many times to fit model parameters to measurements from experiments. Execution times for the fitting code (discussed in more detail in Chapter 5) were often 24 to 72 hr, and it had to be run repeatedly as different reaction mechanisms were investigated. Without these speed improvements, a single run of the fitting code would have taken more than six months to complete.

Parameter	Value	Notes
'Verbose'	'false'	Display results of each iteration
'ScaledStepTol'	10^{-15}	Min Newton step size (too small \rightarrow local min)
'LinearSolver'	'band'	Assume banded Jacobian
'LowerBwidth'	359	Jacobian lower bandwidth
'UpperBwidth'	359	Jacobian upper bandwidth
'Constraints'	vector	Some elements '0' \rightarrow none Some elements '1' \rightarrow ≥ 0
'MaxNumBetaFails'	50	Max number of poor convergence failures
'MaxNewtonStep'	10^9	Max size of Newton step
'MaxNumIter'	500	Limit on non-linear iterations
'FuncNormTol'	10^{-3}	Stopping criterion for residual 2-norm

2.2 Governing Equations and Boundary Conditions

The residual for each state variable is the error in the associated governing equation. A governing equation must be provided for each state variable: conservation of mass (associated with pressure), conservation of momentum (associated with velocity), conservation of species (associated with mass fraction) and electroneutrality (associated with electric potential). Boundary conditions are applied at the inlets, outlets, electrodes, electrode interfaces and membrane interfaces. Some variables have one simple boundary condition and others have multiple mixed boundary conditions, which are linked across interfaces by the related fluxes.

2.2.1 Pressure Residuals: Conservation of Mass

Pressure is associated with mass conservation because mass fluxes are predominantly driven by pressure gradients. At steady state, mass conservation in the channels is the sum of mass fluxes across the boundaries of a numerical cell, therefore the residuals associated with mass conservation are:

$$R_{ch}^P = -\nabla \cdot \vec{J} \quad \text{Eq. 2.3}$$

Mass conservation is also used to solve for pressure at the membrane interfaces, where two fluxes in the y -direction are involved (one mass flux from the channel and one from the membrane):

$$R_{m,int}^P = \vec{J}_{ch} - \vec{J}_{mem} \quad \text{Eq. 2.4}$$

The x -direction pressure boundary condition is constant pressure at the outlets ($P = 100$ kPa). The y -direction pressure boundary conditions are zero pressure gradient at the electrode ($\partial P/\partial y = 0$) in recognition of the zero net mass flux there.

2.2.2 Mass Fraction Residuals: Conservation of Species

Species concentrations are derived from species conservation, which is written in terms of mass fraction to simplify the mass and momentum conservation equations. The concentration of species k is calculated from mass fraction as $C_k = \rho Y_k/W_k$. The species conservation equation is the sum of species flux and storage terms: $0 = \nabla \cdot \vec{J}_k + \partial(\rho Y_k)/\partial t$. The storage term can be rearranged to isolate $\partial Y_k/\partial t$ on the left hand side, which should be zero at steady state and becomes the mass fraction residual. The mass storage term $R_{ch}^P = \partial \rho/\partial t$ is zero at steady state, but it is retained to improve solver convergence by maintaining the link between species and mass conservation. The mass fraction residual for species k in the channel is therefore:

$$R_{k,ch}^{MF} = \frac{1}{\rho} (-\nabla \cdot \vec{J}_k - Y_k R_{ch}^P) \quad \text{Eq. 2.5}$$

Species conservation is also used to find mass fraction residuals at the electrode and membrane interfaces, but it becomes a flux match because the interfaces have only two species fluxes and no mass storage term. At the electrode interfaces there is one flux from the channel and one flux from the electrode (due to reactions there) for each species k :

$$R_{k,e_i}^{MF} = \vec{J}_{k,ch} - \vec{J}_{k,rxn} \quad \text{Eq. 2.6}$$

Mass fraction residuals at the membrane interfaces involve one flux from the channel and one through the membrane for each species k :

$$R_{k,m_i}^{MF} = \vec{J}_{k,ch} - \vec{J}_{k,mem} \quad \text{Eq. 2.7}$$

The x -direction mass fraction boundary conditions are specified at the inlet for each species k based on the fuel and oxidizer concentrations. The y -direction mass fraction boundary conditions are the mass fractions at the electrode and membrane interfaces dictated by the respective flux matches. Ultimately, the mass fractions on each side of the membrane are related by the membrane species fluxes, and mass fractions at the electrodes are related to the electrode potential boundary conditions by the reaction rates.

2.2.3 Velocity Residuals: Conservation of Momentum

Momentum conservation in the channels can be written as the sum of advection, pressure, diffusion and electric body force terms:

$$\frac{\partial(\rho\vec{v})}{\partial t} = -\nabla\rho\vec{v} \cdot \vec{v} - \nabla P + \mu\nabla^2\vec{v} - \rho_c\nabla\phi \quad \text{Eq. 2.8}$$

The electric body force term $\rho_c\nabla\phi$ accounts for the influence of the electric potential gradient on ions in solution. Taking only components involved in the x -direction momentum balance and rearranging to obtain $\partial v_x/\partial t$ (which should be zero at steady state) yields the x -direction velocity residual in the channels:

$$R_{ch}^{xM} = \frac{1}{\rho} \left[-\frac{\partial}{\partial x} (\rho v_x |v_x|) - \frac{\partial}{\partial y} (\rho v_x v_y) - \frac{\partial P}{\partial x} + \mu \left(\frac{\partial^2 v_x}{\partial x^2} + \frac{\partial^2 v_x}{\partial y^2} \right) - \rho_c \frac{\partial \phi}{\partial x} - v_x R_{ch}^P \right] \quad \text{Eq. 2.9}$$

The mass conservation storage term R_{ch}^P is retained in Eq. 2.9 for the same reasons as in Eq. 2.5. The y -direction velocity residuals R_{ch}^{yM} have the same form as Eq. 2.9:

$$R_{ch}^{yM} = \frac{1}{\rho} \left[-\frac{\partial}{\partial x} (\rho v_y v_x) - \frac{\partial}{\partial y} (\rho v_y |v_y|) - \frac{\partial P}{\partial y} + \mu \left(\frac{\partial^2 v_y}{\partial x^2} + \frac{\partial^2 v_y}{\partial y^2} \right) - \rho_c \frac{\partial \phi}{\partial y} - v_y R_{ch}^P \right] \quad \text{Eq. 2.10}$$

Velocity and pressure gradients in the x -direction are estimated by upwind differencing because transport in the x -direction is dominated by advection, and upstream state variables have more influence over advection than downstream state variables [76]. Velocity and pressure gradients in the y -direction are estimated by center differencing because they are small and transport in the y -direction is dominated by diffusion and migration. Velocity boundary conditions were assigned at the channel inlets, outlets and interfaces as shown in Table 2-2.

The dynamic viscosity μ is assumed constant and dictated by C_{NaOH} for the fuel solution and $C_{H_2SO_4}$ for the oxidizer solution. For 2 M NaOH at 298 K, $\mu \approx 1.41 \times 10^{-3}$ Pa·s [27]. For 0.5 M H_2SO_4 at 298 K, μ is nearly that of water, so the value for water was used (8.90×10^{-4} Pa·s).

Table 2-2. Velocity boundary conditions in the model.

Boundary	Description	Boundary Condition(s)
Channel Inlets	Fully developed with specified volumetric flow rate	$v_x = 1.5\dot{V}(1 - (2y/h - 1)^2)$ $v_y = 0$
Channel Outlets	Fully developed	$\partial v_x / \partial x = 0, \quad \partial v_y / \partial x = 0$
Electrode Interface	No slip condition	$v_x = 0$
	Impermeable boundary	$v_y = 0$
Membrane Interface	No slip condition	$v_x = 0$
	No gradient in membrane	$\partial v_y / \partial y = 0$

2.2.4 Electric Potential Residuals: Electroneutrality

As discussed in Chapter 1, the electric potential in an electrolyte solution is related to the net charge density by Poisson's electrostatic equation (Eq. 1.6). The small permittivity of free space ($\epsilon_0 = 8.85419 \times 10^{-12} \text{ F}\cdot\text{m}^{-1}$) makes the electric field gradient sensitive to any deviation from electroneutrality, and in real systems such deviations only occur in electrochemical double layers. Elsewhere the net charge density can be assumed to be zero [77]. Some models [72] have solved Eq. 1.6 in order to resolve double layers. In this work, the double layers are considered part of the interfaces, and so the electric potential is found by enforcing electroneutrality. The electric potential residual is equal to the net charge density (i.e. deviation from electroneutrality) as given by Eq. 2.11, which is solved to find the electric potential in the channels and at the interfaces.

$$R^\phi = F \sum_k z_k C_k \quad \text{Eq. 2.11}$$

Charge conservation implies that the total anode current must equal the total cathode current at steady state. The electroneutrality condition ensures the total electrode currents are equal by conserving charge throughout the model domain.

The x -direction boundary conditions on electric potential are zero electric field at the inlets and outlets ($\partial\phi/\partial x = 0$) and the y -direction boundary conditions are the electrode potentials. The anode electric potential is zero to establish a reference for the system, and the cathode electric potential is the specified cell voltage.

2.3 Species, Mass and Charge Fluxes in the Channels

The species, mass and charge fluxes crossing scalar numerical cell boundaries must be calculated in order to evaluate the related conservation equation residuals. Solute species mole fluxes in the channel are described by the Nernst-Planck equation (Eq. 1.20), which includes diffusion, migration and advection.

Since diffusion cannot produce a net mass flux, the water diffusion mass flux is made equal to the negative sum of solute diffusion mass fluxes. Water has no net charge, so the migration term is dropped from the total water mole flux in Eq. 2.12.

$$\vec{J}_{H_2O} = \frac{1}{W_{H_2O}} \left(\sum_{k \neq H_2O} (W_k D_k \nabla C_k) + \rho Y_{H_2O} \vec{v} \right) \quad \text{Eq. 2.12}$$

The total mass flux is the molecular mass weighted sum of the mole fluxes:

$$\vec{J} = \sum_k W_k \vec{J}_k \quad \text{Eq. 2.13}$$

The total charge flux is a charge-weighted sum of the mole fluxes:

$$\vec{J}_c = F \sum_k z_k \vec{J}_k \quad \text{Eq. 2.14}$$

The mass, species and charge flux equations require concentration and electric potential gradients at the discretization boundaries. These gradients are estimated by center differencing because it tends to be more numerically stable for diffusion and migration [76]. Table 2-3 lists the transport and thermophysical properties appearing in the flux equations.

Table 2-3. Diffusivities and apparent molar volumes of fuel and oxidizer solutes in H₂O at infinite dilution and 298 K.

Species	Diffusivity [m ² s ⁻¹ x 10 ⁹]	Apparent molar volume [m ³ kmol ⁻¹ x 10 ³]
BH ₄ ⁻	2.42 [61]	-5.83 ^a
BO ₂ ⁻	0.814 [78]	-14.5 [74]
H ⁺	9.312 [77]	0 [74]
H ₂ O ₂	1.19 [75]	22.17 [75]
Na ⁺	1.334 [77]	-1.11 [74]
OH ⁻	5.260 [79]	-4.18 [74]
SO ₄ ⁻²	0.625 [80]	24.8 [81]

^a Value estimated to be smaller than BO₂⁻ by the diffusivity ratio, because it was unavailable.

2.4 Species, Mass and Charge Fluxes Through the Membrane

A 1-D sub-model was created to calculate the Na⁺ and H₂O mole fluxes across a Nafion membrane at each point down the channel. Other species are not addressed by the model, which implies that the concentrations of anions in the fuel and oxidizer solutions are smaller than C_{SO₃⁻} in the membrane (see §1.5.6). Excluding other

species may introduce error in the calculation of $\Delta\phi_{mem}$, but the focus of this model is an operating DBFC, not prediction of the open circuit cell voltage. As discussed in §1.5.6, the rates of transport of other species through the membrane are greatly decreased once the cell current density exceeds zero. Table 2-4 lists transport parameters in the membrane sub-model, and Figure 2.6 illustrates the membrane sub-model schematically.

The membrane is assumed to be fully hydrated, which in the Na^+ form means $\lambda = 22$ and $n_{d,\text{Na}^+} = 9.2 \text{ kmol H}_2\text{O} / \text{kmol Na}^+$ [64]. The full hydration assumption is reasonable given that both sides of the membrane are in contact with aqueous solutions, understanding that in a real DBFC λ may be less than 22 due to the osmotic gradients between the membrane and hypertonic solutions [82].

The fully hydrated, thin Nafion membranes (N117: 208 μm and N115: 161 μm [83]) enabled two simplifications of the membrane model. First, gradients in the membrane ($\nabla\phi, \nabla C_k, \nabla P$) were assumed to be linear and the membrane is not discretized in the y -direction. The value of each gradient is the difference between the values at each membrane-solution interface, divided by the membrane thickness. Second, transport through the membrane in the x -direction was omitted as any x -direction flows through the membrane will be insignificant compared to x -direction flows in the fuel and oxidizer solutions near the membrane.

Na^+ is driven through the membrane by migration and diffusion. Thus, the net Na^+ mole flux from fuel channel to oxidizer channel is given by Eq. 2.15.

$$\vec{J}_{\text{Na},mem} = -D_{\text{Na}} \frac{dC_{\text{Na}}}{dy} - z_{\text{Na}} u_{\text{Na}} F C_{\text{Na}} \frac{d\phi}{dy} \quad \text{Eq. 2.15}$$

To maintain electroneutrality in the membrane, the concentration of Na^+ must be equal to the concentration of SO_3^- groups: $C_{\text{Na}^+} = C_{\text{SO}_3^-}$. Since $C_{\text{SO}_3^-}$ in the membrane is uniform, the Na^+ concentration gradient must be zero: $dC_{\text{Na}^+}/dy = 0$. Diffusion of Na^+ through the membrane in response to the difference in concentration ΔC_{Na^+} across the membrane does take place, however. The concentration gradient term in Eq. 2.15 captures the contribution ΔC_{Na^+} makes to $\Delta\tilde{\mu}_{\text{Na}^+}$ by calculating dC_{Na^+}/dy based on the concentrations at the membrane-solution interfaces. $\Delta\tilde{\mu}_{\text{Na}^+}$ is a more accurate description of the driving force for Na^+ transport through the membrane.

Table 2-4. Summary of thermophysical and transport properties in fully hydrated Nafion 115 in the Na^+ form at 298 K

Property	Value
Na^+ mobility, $u_{\text{Na}^+,mem}$	$2.7 \times 10^{-8} \text{ m}^2 \cdot \text{V}^{-1} \cdot \text{s}^{-1}$ [67]
H_2O permeability, $p_{\text{H}_2\text{O}}$	$1.7 \times 10^{-14} \text{ m} \cdot \text{Pa}^{-1} \cdot \text{s}^{-1}$ [64]
Na^+ concentration, $C_{\text{Na}^+} = C_{\text{SO}_3^-}$	$1.13 \text{ kmol} \cdot \text{m}^{-3}$ [84]
Na^+ - H_2O electro-osmotic drag coefficient in fully hydrated Nafion in the Na^+ form, n_d	$9.2 \text{ kmol H}_2\text{O} / \text{kmol Na}^+$ [67]

Water crosses the membrane due to diffusion, permeation and electro-osmotic drag induced by the Na^+ flux. Water diffusion is neglected because $C_{\text{H}_2\text{O}}$ is nearly the same on both sides of the membrane. Thus, the net water flux is determined by the permeation and electro-osmotic drag terms.

$$\vec{J}_{\text{H}_2\text{O},mem} = -p_{\text{H}_2\text{O}}\Delta P \frac{\rho_{\text{H}_2\text{O}}}{W_{\text{H}_2\text{O}}} + n_{d,\text{Na}^+}\vec{J}_{\text{Na},mem} \quad \text{Eq. 2.16}$$

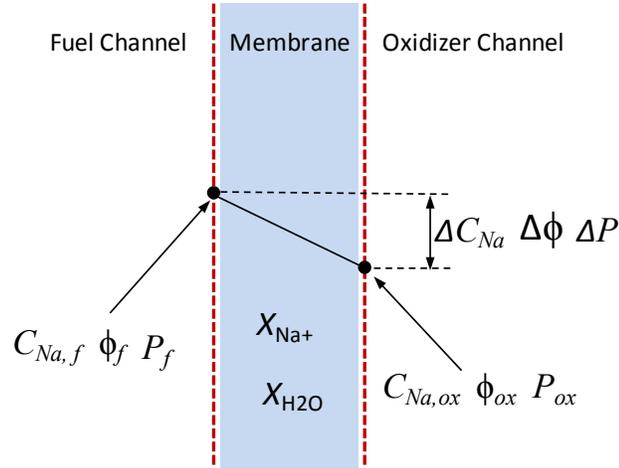


Figure 2.6. Illustration of state variable locations and gradient calculation in the membrane sub-model.

2.5 Species and Charge Fluxes at the Electrodes

Multiple charge transfer reactions can take place at an electrode simultaneously; the rates are estimated by rate expressions of the form in Eq. 1.11. The rates of non-charge transfer reactions, such as R 1.4, are evaluated using the same expression with $n_e = 0$. The net flux of each species at an electrode is the sum of fluxes due to all reactions taking place, multiplied by a roughness factor ℓ which accounts for the ratio of electrochemical to geometric electrode surface area. With r_q the rate of reaction q and $v_{k,q}$ the stoichiometry of species k in reaction q , the net flux of k due to reactions at the electrode is given as Eq. 2.17.

$$\vec{J}_{k,rxn} = \ell \sum_q r_q v_{k,q} \quad \text{Eq. 2.17}$$

The net current density at the electrode is found by summing the current density contributions from each species:

$$i_{net} = F \sum_k \vec{J}_{k,rxn} z_k \quad \text{Eq. 2.18}$$

The possibility of surface adsorbed species participating in the electrode reactions is explored in Chapter 5. The surface concentration may be calculated by assuming a Langmuir adsorption isotherm, for which the fraction of surface sites occupied by species k , θ_k , is related to the concentration in solution (see Eq. 2.19) and a constant β_k . This approach presumes that the rate of adsorption and desorption for species k is fast in comparison to the rates of reactions consuming or producing k . The Langmuir isotherm assumes that adsorbed species do not interact, thus the activation energy barrier to adsorption and desorption depends solely on the change in free energy (electrochemical potential) of adsorbing or desorbing species (see Eq. 2.20). Since the change in electrochemical potential includes the change in ion electrostatic potential energy as it moves from the electrode interface to the electrode surface through the double layer, θ_k will shift with electrode potential. For example, a cation will adsorb more readily on a less positive electrode because the electrostatic component of the activation energy barrier will be smaller. The surface site fraction of species k calculated via Eq. 2.19 and Eq. 2.20 replaces the concentration of k in the rate equation. In this sense, θ_k is treated as a surface activity of species k .

$$\theta_k = \frac{\beta_k C_k}{1 + \beta_k C_k} \quad \text{Eq. 2.19}$$

$$\beta_k = e^{-\Delta\tilde{\mu}_k/RT} \quad \text{Eq. 2.20}$$

An adsorption isotherm permits implicit calculation of adsorption and desorption rates by assuming surface species are in equilibrium with the aqueous species, but the rates can also be calculated explicitly. At steady state, the net rate of species k addition to the surface must equal the rate of loss, with the net rate being a result of adsorption/desorption reactions and surface reactions. The sum over production and consumption rates due to all reactions q must therefore be zero, and the residual associated with the surface site fraction of each species k is given by Eq. 2.21.

$$R_k^{SF} = \sum_q r_q \nu_{k,q} \quad \text{Eq. 2.21}$$

2.6 Composition Equation of State

The fuel and oxidizer solutions are incompressible, but an equation of state is still needed to relate solution mass density to composition. A state equation (Eq. 2.22) relates solution mass density to mass fraction by accounting for the apparent molar volume of each solute.

$$\rho = \frac{\rho_{H_2O}}{Y_{H_2O} + \rho_{H_2O} \sum_k (\delta_k Y_k / W_k)} \quad \text{Eq. 2.22}$$

The density of pure water, ρ_{H_2O} , varies with temperature via a polynomial fit to empirical data at 100 kPa.

Chapter 3: An Analysis of Ideal DBFC Performance

3.1 *Goals and Approach*

While there has been significant work on the electrochemistry of DBFCs, practical DBFC cell design and operating spaces have not been explored as thoroughly. Experiments on fundamental kinetics as well as electrode geometry have provided insight into what is feasible for DBFCs, but relationships between cell design and performance are not understood. The range of reactant and product concentrations along the flow path of a practical cell with substantial reactant utilization can have a significant impact on reaction overpotentials and transport limitations, and thus overall cell efficiencies and power densities. Carefully constructed numerical models of DBFCs can explore cell configurations and designs that provide the highest power densities and the most effective conversion of fuel and oxidizer to useful energy (i.e., efficiency). A model can also provide state information that is useful for understanding how the cell operates, but is otherwise difficult to measure. This chapter presents results from a 2-D steady state finite volume model which explore how cell geometry and operating conditions impact DBFC performance under conditions which favor R 1.2 and R 1.3.

The modeled system is ideal in that only R 1.2 and R 1.3 take place, and the membrane is permeable only to Na^+ and water. Realistically, improved electrocatalysts and membranes are unlikely to eliminate R 1.4 and R 1.5, or the transport of species other than Na^+ through the membrane. The results presented here

represent the performance of an ideal DBFC, which realistic DBFC performance may approach as the technology improves.

The rates of R 1.2 and R 1.3 were written in the form of Eq. 1.11, making use of Eq. 1.13 to write them in terms of only one rate constant. This approach ensured thermodynamic consistency, i.e. the net rate would be zero under standard conditions (all $\alpha_k=1$) and electrode potential equal to $\Delta\phi'$. The concentrations of species in excess (OH^- at the anode, H^+ at the cathode and H_2O at both electrodes) were neglected as they have little effect on the rates. The rates of R 1.2 and R 1.3 are given as Eq. 3.1 and Eq. 3.2.

$$r_{R1.2} = k_{a,R1.2} \left\{ C_{\text{BH}_4^-} e^{\beta_{a,R1.2} f \Delta\phi} - e^{f E_{R1.2}^0} C_{\text{BO}_2^-} e^{-(1-\beta_{a,R1.2}) f \Delta\phi} \right\} \quad \text{Eq. 3.1}$$

$$r_{R1.3} = k_{c,R1.3} \left\{ e^{-f E_{R1.3}^0} e^{\beta_{a,R1.3} f \Delta\phi} - C_{\text{H}_2\text{O}_2} e^{-(1-\beta_{a,R1.3}) f \Delta\phi} \right\} \quad \text{Eq. 3.2}$$

This study consists of a baseline case and 12 alternative cases in which one parameter is altered (see Table 3-1) to identify the effects it has on DBFC performance. Many parameters remained the same in all cases (see Table 3-2). The model was run repeatedly for each case for a range of cell voltages. An initial solution at a cell voltage of 3.4 V was calculated, and the cell voltage was subsequently decreased in 0.1 V steps until it reached the transport-limited current density. Each run yielded values for all state variables in the model domain and post processing provided the mass, momentum and charge fluxes (i.e. current density). The state variables and fluxes were used to estimate performance metrics such as voltage efficiency, power density and fuel utilization.

Baseline values for the channel dimensions, membrane thickness and inlet flow rates were based on typical values appearing in published experimental studies. The inlet BH_4^- concentrations were chosen to operate in a regime where complete borohydride oxidation (R 1.2) is likely to dominate the anode reactions (at cell voltages other than open circuit). Fast forward reaction rate constants ($k_{R1.2,a}$ at the anode and $k_{R1.3,c}$ at the cathode) were chosen to model cell performance with an advanced electrocatalyst and emphasize the influence of transport on cell performance.

Table 3-1. Baseline case and parameter variations

Parameter	Baseline	Variations from Baseline
Anode inlet [NaBH_4]	0.3 M	0.1, 0.2, 0.4, 0.5 M
Channel depth	1 mm	0.50, 0.75 mm
Membrane thickness	145 μm	1.45, 72.50 μm
Inlet fuel flow rate	60 mL min^{-1}	15, 30 mL min^{-1}
Forward reaction rate constants	$10^6 \text{ m}^4 \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$	$10^0, 10^3 \text{ m}^4 \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$

Table 3-2. Parameters common to all cases

Parameter	Value	Parameter	Value
Anode inlet [NaOH]	4.0 M	Cathode inlet [H_2SO_4]	4.0 M
Anode inlet [NaBO_2]	10^{-6} M	Cathode inlet [H_2O_2]	4.0 M
Channel width	10 mm	Oxidizer inlet flowrate	60 $\text{mL} \cdot \text{min}^{-1}$
Temperature	298 K		

3.2 Baseline Case

The calculated baseline polarization curve is shown in Figure 3.1. The cell voltages in Figure 3.1 are useful in a relative sense because the model neglects fuel crossover and competing electrochemical reactions, but the slopes of the polarization curve in the linear region at intermediate current densities and in the transport-limited

region at the highest current densities are comparable to an actual cell because they are largely dictated by known transport parameters.

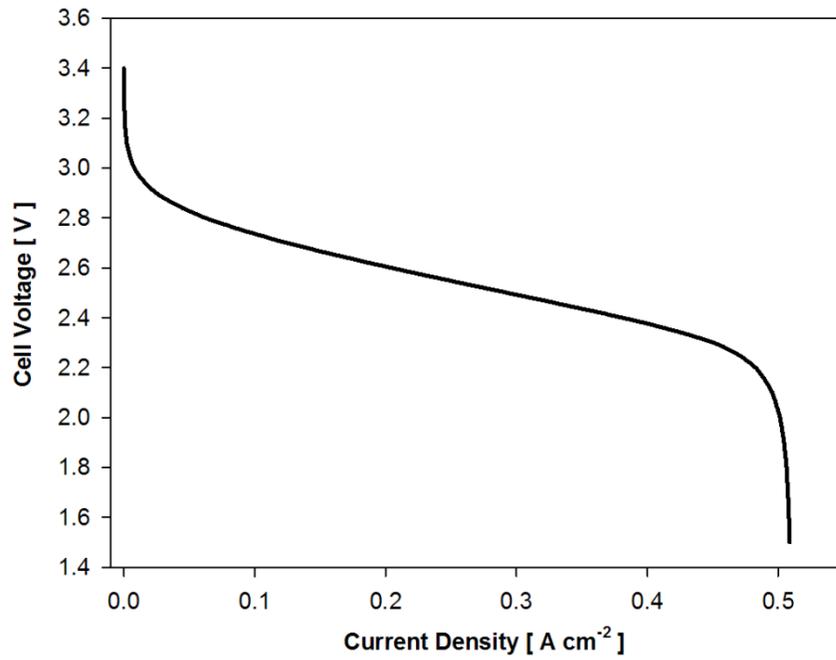


Figure 3.1. Baseline polarization curve showing activation, ohmic and concentration overpotentials.

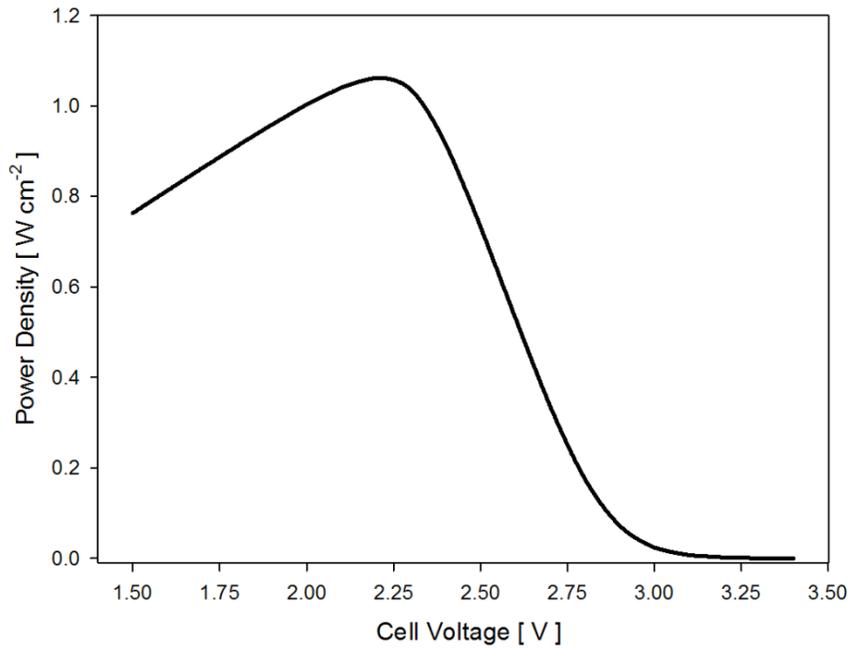


Figure 3.2. Baseline power density vs. cell voltage. Points to the right of 2.2 V offer the best combination of power density and voltage efficiency.

The product of cell voltage and average current density gives an average cell area power density ($\text{W}\cdot\text{m}^{-2}$ of electrolyte membrane), which provides a link to stack (volume) power density ($\text{W}\cdot\text{m}^{-3}$). Plotting power density vs. cell voltage as in Figure 3.2 illustrates the tradeoffs between power density and voltage efficiency. Figure 3.2 shows baseline peak power density occurs at 2.2 V. Operating this DBFC under these operating conditions at cell voltages higher than 2.2 V offers the best combination of power density and efficiency.

The voltage losses due to activation, concentration and ohmic resistance vary along the channel with local concentrations and current density. The magnitude of each loss can be characterized by the associated area specific resistance (ASR). Figure 3.3 plots the variation of the ASRs for each electrode and the membrane along the channel at a cell voltage of 2.2 V. For the baseline reaction rates, the ASRs for concentration in the fuel channel and cathode activation are largest, indicating they contribute most to losses in the fuel cell at this operating point. Both activation and concentration ASRs increase down the channel as concentration boundary layers lower the availability of reactants at the electrodes and depress the actual OCV further from the ideal OCV. Figure 3.3 shows the importance of variation down the channel in a DBFC.

The variation in the anode concentration ASR with distance from the inlet is explained in Figure 3.4, which shows current density as a function of distance from the inlet. BH_4^- is consumed rapidly near the channel inlet but the consumption rate

falls near the outlet. Thus, while channel fuel utilization (Eq. 3.3) increases with channel length, the increased length results in lower power densities. The channel fuel utilization in the case shown in Figure 3.4 is only 4.4%. As seen in Figure 3.3, the fuel channel concentration ASR becomes higher than other resistances with distance along the channel due to the growing thickness in the fuel concentration boundary layer.

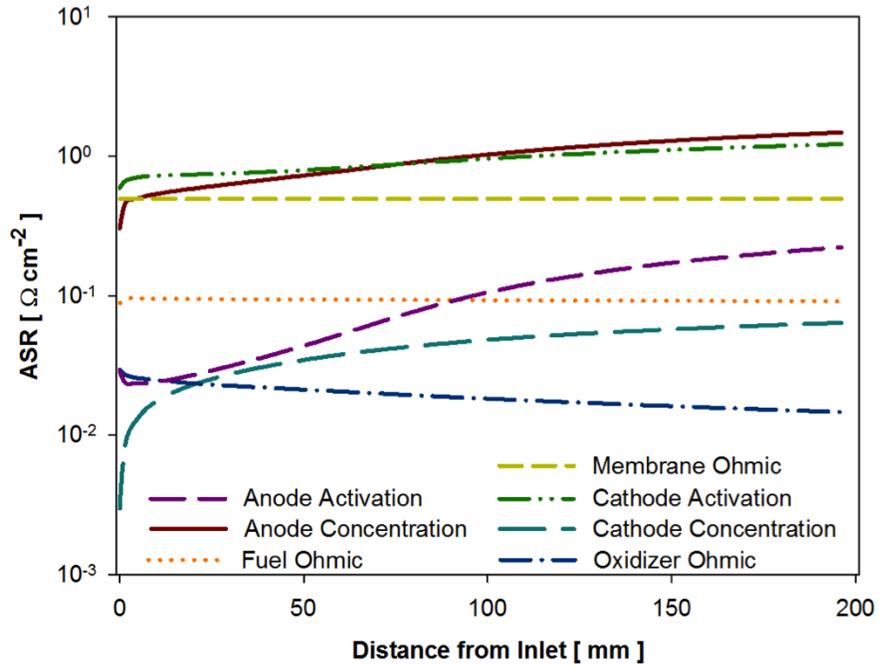


Figure 3.3. Area specific resistance (ASR) associated with each overpotential at points down the channel, in the baseline case at 2.2 V.

$$\eta_{ru} = \frac{\int_0^{L_{cell}} \vec{J}_{BH_4^-}(x) dx}{C_{BH_4^-} \dot{V}} \quad \text{Eq. 3.3}$$

Transport through this boundary layer in the y -direction is complicated because advection, diffusion and migration all generate BH_4^- fluxes in the y -direction. Figure

3.5 shows anode channel BH_4^- fluxes in the y -direction, mid-way between the inlet and outlet, at a cell voltage of 2.2 V. Transport at the anode surface is dominated by diffusion because the concentration of BH_4^- there is nearly zero, but migration plays a significant role in the center of the channel where the concentration is higher and the electric field drives BH_4^- toward the anode.

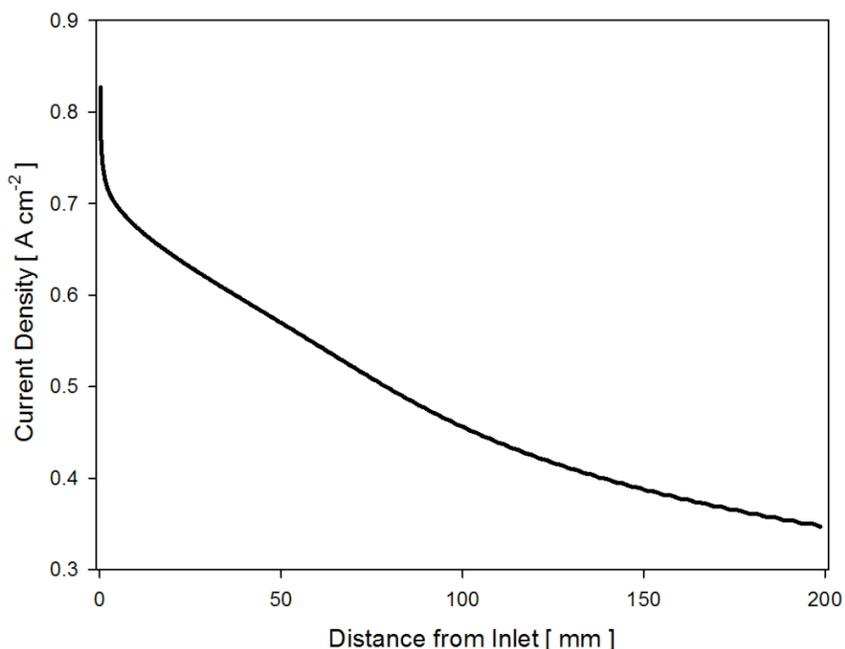


Figure 3.4. Baseline current density at 2.2 V at points down the channel, which correlates with the rate of reactant consumption. Anode and cathode current densities at each point are the same to within 0.01%.

Transport of other species also impacts the DBFC performance. In particular, transport of Na^+ and H_2O through the membrane produces an advection flux that increases with current density. In the channel, this advection flux carries all species, including BH_4^- as shown in Figure 3.5. Since this model assumes only Na^+ and H_2O pass through the membrane, the net flux of BH_4^- at the membrane interface must be zero, which can only be satisfied with an elevated BH_4^- concentration at the

membrane interface to drive diffusion and migration fluxes which oppose the advection flux.

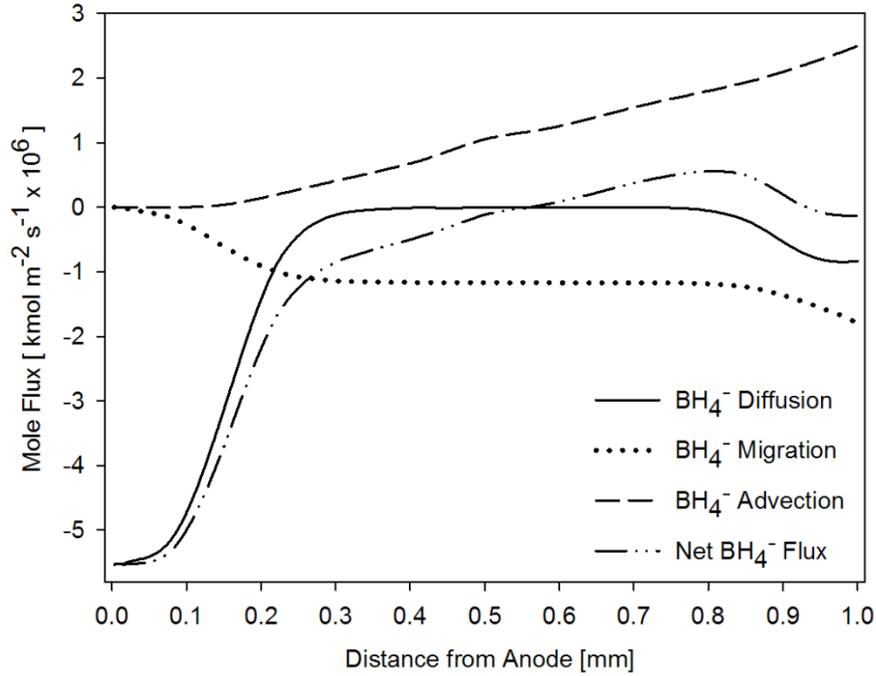


Figure 3.5. Mole fluxes of BH_4^- in the y -direction, half way down the anode channel, as a profile from anode to membrane. Positive fluxes are toward the membrane and negative fluxes are toward the anode. Results for the baseline case at 2.2 V.

Water crosses the membrane at a channel-averaged rate of $8.3 \times 10^{-3} \text{ kg} \cdot \text{m}^{-2} \cdot \text{s}^{-1}$ in the baseline case at 2.2 V, which is 14.6 times larger than the rate of water production at the anode. Large water crossover rates have been observed in experiments [2] and can be problematic for systems with large recirculation fractions. The permeation water flux ($5.5 \times 10^{-11} \text{ kg} \cdot \text{m}^{-2} \cdot \text{s}^{-1}$) is insignificant due to the small ($\sim 5 \text{ Pa}$) pressure difference across the membrane. The pressure difference across the membrane would have to be impractically high for the permeation flux to offset the electro-osmotic drag flux.

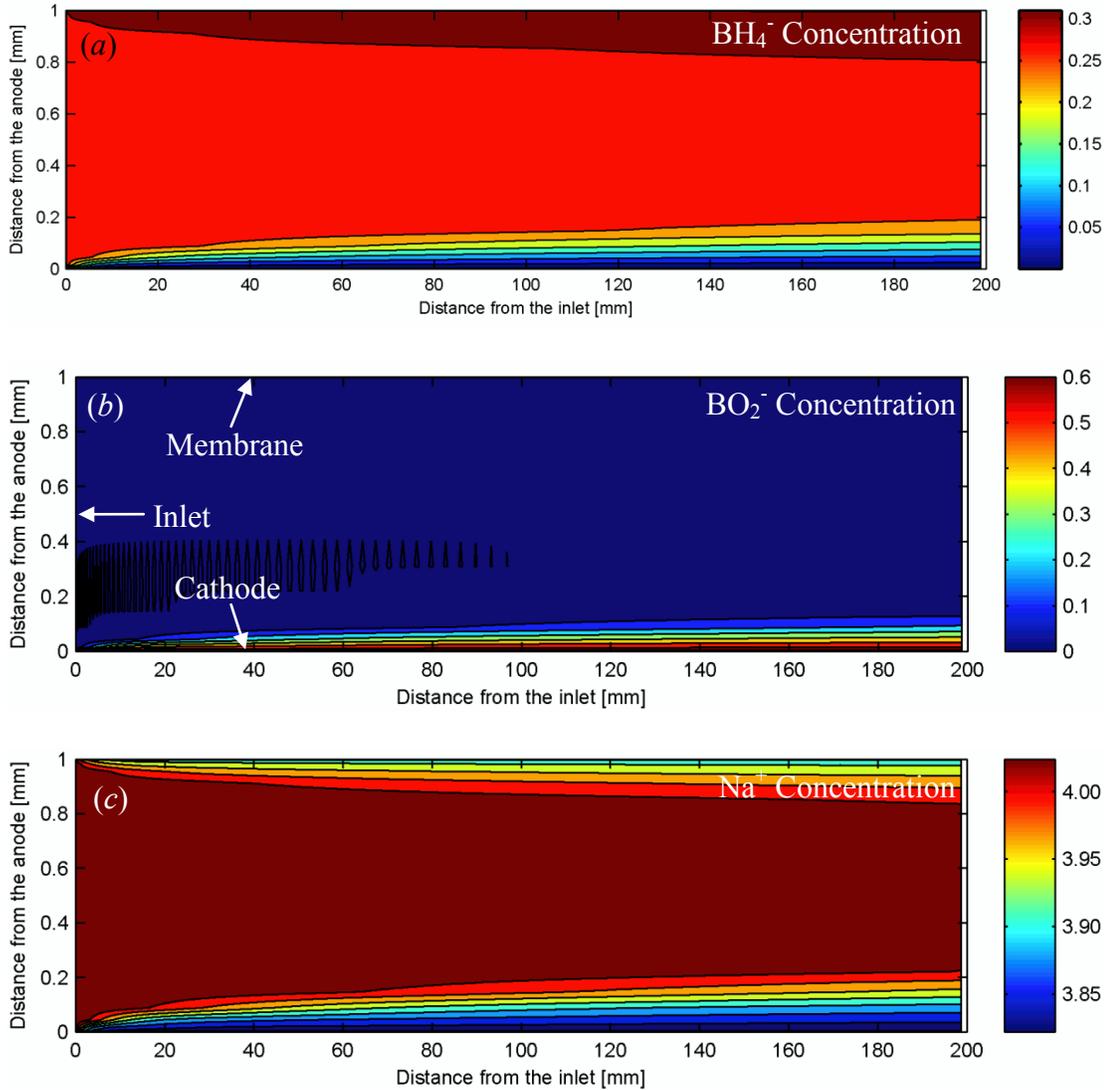


Figure 3.6. Contour plots of (a) BH_4^- , (b) BO_2^- and (c) Na^+ concentration ($\text{kmol}\cdot\text{m}^{-3}$) in the fuel channel for the baseline case at 2.2 V (peak power density). Channel boundaries are the same in each plot.

Contour plots of species concentration in the channel (Figure 3.6) provide insight into cell operation. For example, the compact BH_4^- and BO_2^- concentration boundary layers near the anode explain the low fuel utilization. Similarly, the Na^+ concentration boundary layer at the anode can be explained by diffusion, migration and advection interacting with the electroneutrality condition. The electric field drives cations away from the anode, but the Na^+ flux must be zero there because the

wall is impermeable and Na^+ does not participate in the reaction. The Na^+ concentration near the anode is depressed to create a diffusion flux that opposes the migration flux and makes the net flux zero. Anion (BH_4^- and BO_2^-) concentrations must also decrease to maintain a neutral solution, and the lower anion concentrations change the rates of transport and local OCV.

3.3 Effects of Reaction Rate Constant

For the baseline case, fast forward reaction rate constants were chosen to emphasize transport processes, but catalysts in DBFC experiments are often less active, as discussed in §1.5.5. Slower anode rates were examined to understand how a less active anode catalyst would affect cell performance. Three polarization curves in Figure 3.7 were generated by varying the anode forward reaction rate constant $k_{R1.2,a}$ between 10^0 and $10^6 \text{ m}^4 \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$ with the cathode forward reaction rate constant $k_{R1.3,c} = 10^6 \text{ m}^4 \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$. This range of rate constants corresponds to a shift in activation energy barrier E_{act} of approximately $35 \text{ J} \cdot \text{mol}^{-1}$ (0.35 eV), which should encompass the range of uncertainty for a selected catalyst.

The trend in Figure 3.7 with increasing reaction rate constant (lower E_{act}) indicates performance benefits that may be obtained with the development of more active anode catalysts for the desired reactions. A faster anode reaction rate decreases the anode activation overpotential, but with diminishing returns for $k_{R1.2,a} > 10^6 \text{ m}^4 \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$, beyond which point the anode activation overpotential is much smaller than other overpotentials in the cell.

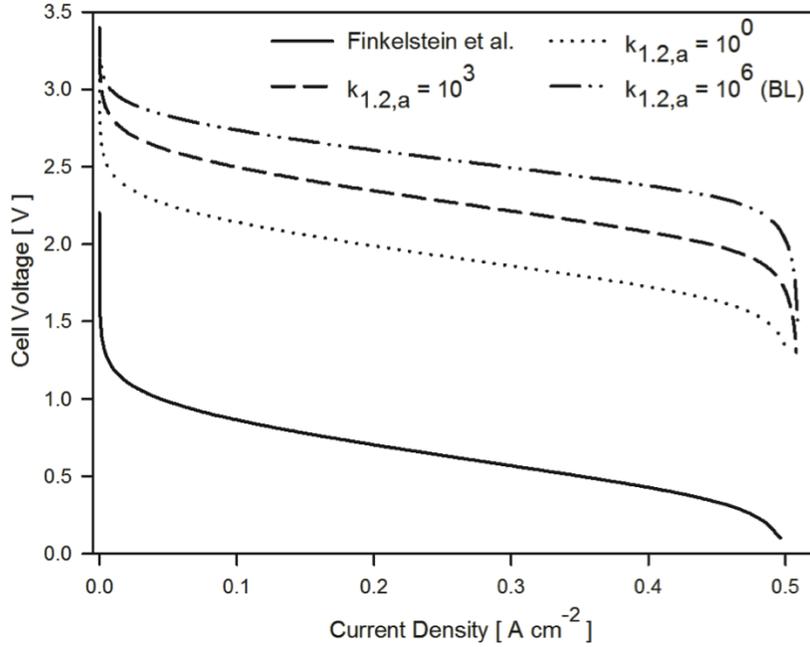


Figure 3.7. Polarization curves with $k_{R1.2,a} = 10^0, 10^3$ and $10^6 \text{ m}^4 \text{ mol}^{-1} \text{ s}^{-1}$ and $k_{R1.3,c} = 10^6 \text{ m}^4 \text{ mol}^{-1} \text{ s}^{-1}$. The bottom curve assumes k_a and k_c from Finkelstein et al. on Pt; anode: $2 \text{ m}^4 \cdot \text{kmol}^{-1} \cdot \text{s}^{-1}$ and cathode: $1.6 \text{ m}^4 \cdot \text{kmol}^{-1} \cdot \text{s}^{-1}$. All parameters, other than rate constants, are the same as in the baseline case, given in Table 3-1.

A fourth polarization curve in Figure 3.7 was generated using rate constants from the literature for BH_4^- oxidation on Pt ($k_{R1.2,a} = 2 \text{ m}^4 \cdot \text{kmol}^{-1} \cdot \text{s}^{-1}$ [39]) and H_2O_2 reduction on Pt ($k_{R1.3,c} = 1.6 \text{ m}^4 \cdot \text{kmol}^{-1} \cdot \text{s}^{-1}$ [62]). This curve shows that using Pt at both electrodes will incur large activation losses, and peak power density will be lower ($0.175 \text{ W} \cdot \text{cm}^{-2}$) compared to the transport dominated baseline case ($1.06 \text{ W} \cdot \text{cm}^{-2}$). For comparison, power densities of $0.05 - 0.50 \text{ W} \cdot \text{cm}^{-2}$ are typical of DBFC experiments, depending on cell design, operating conditions and catalyst selection [25].

Sensitivity to reaction rate was evaluated by changing the forward reaction rates at both electrodes from the baseline values. Comparing ASR plots in Figure 3.3 ($k_{R1.2,a} = k_{R1.3,c} = 10^6 \text{ m}^4 \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$) to Figure 3.8 ($k_{R1.2,a} = k_{R1.3,c} = 10^3 \text{ m}^4 \cdot \text{mol}^{-1} \cdot \text{s}^{-1}$)

shows that the anode activation ASR falls much faster than the cathode activation ASR with increasing forward rate constants. For the conditions here, the anode activation overpotential is therefore more sensitive to forward reaction rate constant.

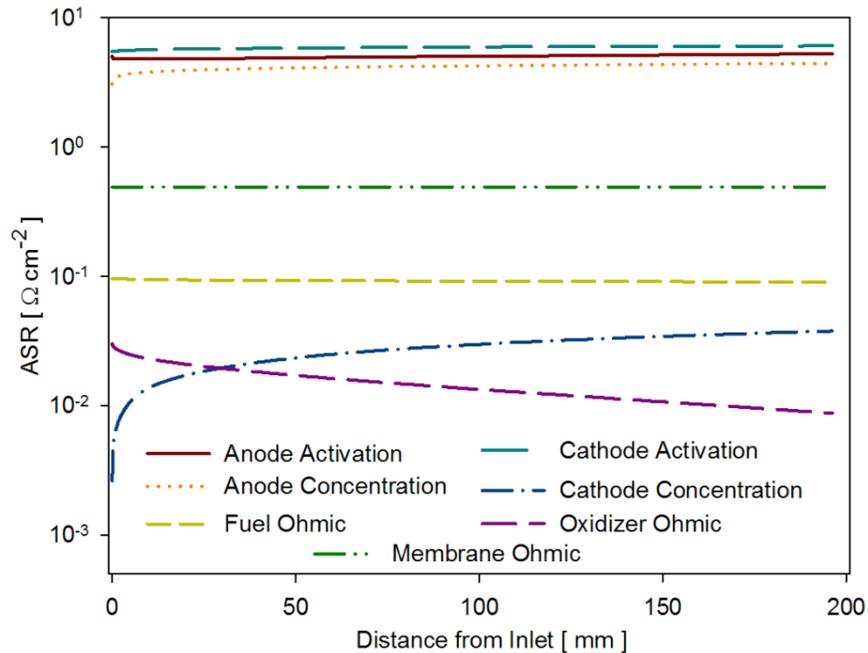


Figure 3.8. Area specific resistance at points down the channel, for the case with forward reaction rates set to 10^3 and cell voltage of 2.2 V. Aside from reaction rate constants, all parameters are the same as in the baseline case.

3.4 Effects of Cell Geometry

For the cell topology illustrated in Figure 1.1, two dimensions pertinent to cell performance are channel and membrane thickness. The electrolyte solutions (in the channels) and the membrane each have an electrical resistivity determined by the mobilities of charged species in the respective phase. The electrical resistivity incurs an ohmic loss which is reduced with narrower channels. Furthermore, narrower channels result in a steeper voltage gradients in the y -direction, which produce larger

migration fluxes. The larger migration fluxes can augment electrode reactions and transport through the membrane.

Figure 3.9 plots power density vs. cell potential for channels 0.50, 0.75 and 1.00 mm deep. The mean fuel solution inlet velocity was held fixed at $0.1 \text{ m}\cdot\text{s}^{-1}$ in each case. Shallower channels clearly provide higher power density with little sacrifice of voltage efficiency. This benefit is amplified further when considering volumetric power density because the shallower channels enable more cells to fit in a given volume. A further advantage of shallower channels is higher channel fuel utilization. With shallower channels, the concentration boundary layer occupies a larger fraction of the channel and therefore less fuel flows through the channel unreacted. Other performance metrics are listed in Table 3-3.

Table 3-3. Performance metrics with respect to channel depth with all other parameters at the baseline case

Parameter	Channel Depth [mm]		
	0.50	0.75	1.00
Peak power density [$\text{W}\cdot\text{cm}^{-2}$]	1.28	1.15	1.06
Cell voltage @ peak power density [V]	2.16	2.19	2.21
Fuel utilization @ peak power density [%]	10.2	6.05	4.14
Power density @ 2.5 V [$\text{W}\cdot\text{cm}^{-2}$]	0.82	0.77	0.73

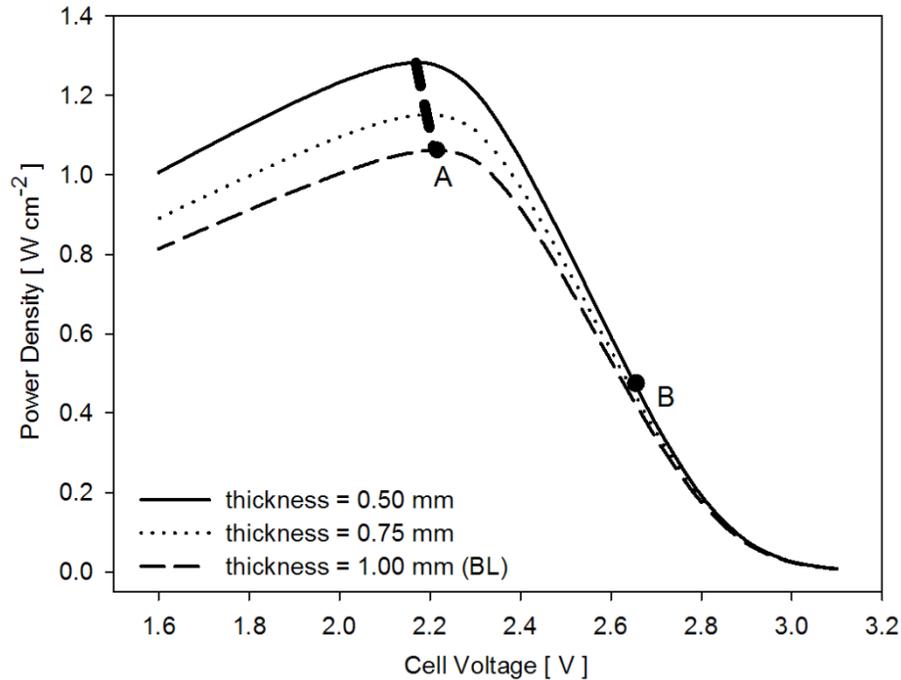


Figure 3.9. Average power density vs. cell potential for channels 0.50, 0.75 and 1.00 mm deep. All other parameters as in the baseline case, labeled “BL”. The bold dashed curve is drawn through the peak power density for each channel depth. The mean velocity of fuel solution entering the fuel channel was fixed at $0.1 \text{ m}\cdot\text{s}^{-1}$.

Most of the ohmic voltage loss between the two electrodes occurs in the membrane as shown in Figure 3.10, with the losses being highest near the channel inlet where the current densities are highest. This is due to the relatively low diffusivity of Na^+ in Nafion ($6.93 \times 10^{-10} \text{ m}^2\cdot\text{s}^{-1}$ vs. $1.33 \times 10^{-9} \text{ m}^2\cdot\text{s}^{-1}$ in water at infinite dilution). A thinner membrane mitigates these losses and improves the cell voltage and power density as illustrated in Figure 3.11, which plots polarization and power density curves for three membrane thicknesses. Reducing membrane thickness from 145 to $1.45 \text{ }\mu\text{m}$ (equivalent to increasing membrane conductivity by 100 times) raises the cell voltage at $0.40 \text{ A}\cdot\text{cm}^{-2}$ from 2.375 V in the baseline case to 2.570 V. The very thin $1.45 \text{ }\mu\text{m}$ -thick membrane accounts for only 3.8 – 4.6% of the total

ohmic losses in the cell (depending distance from the inlets). Of course, the risk of fuel crossover not considered in this model would become much more significant with such a thin membrane and would need to be included in a final analysis. Performance metrics as a function of membrane thickness without including the impact of fuel crossover are listed in Table 3-4.

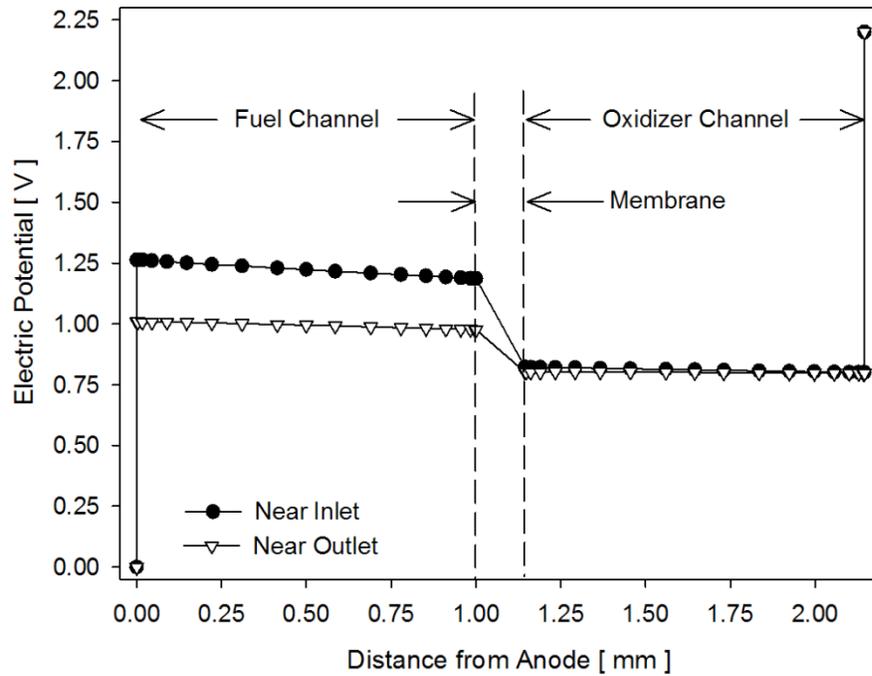


Figure 3.10. Electric potential profiles across the cell from anode to cathode, near the inlet and near the outlet. The membrane clearly makes the largest contribution to the total ohmic losses, and the decrease in ohmic losses from inlet to outlet is related to the decrease in current density. Data in this figure are from the baseline at 2.2 V.

Table 3-4. Performance metrics with respect to membrane thickness with all other parameters at the baseline case

Parameter	Membrane Thickness [μm]				
	1.45	35.25	72.50	108.75	145.00
Peak power density [$\text{W}\cdot\text{cm}^{-2}$]	1.21	1.17	1.13	1.10	1.06
Cell voltage @ peak power density [V]	2.43	2.37	2.32	2.26	2.21
Fuel utilization @ peak power density [%]	4.26	4.24	4.19	4.17	4.14
Power density @ 2.5 V [$\text{W}\cdot\text{cm}^{-2}$]	1.18	1.07	0.94	0.82	0.73

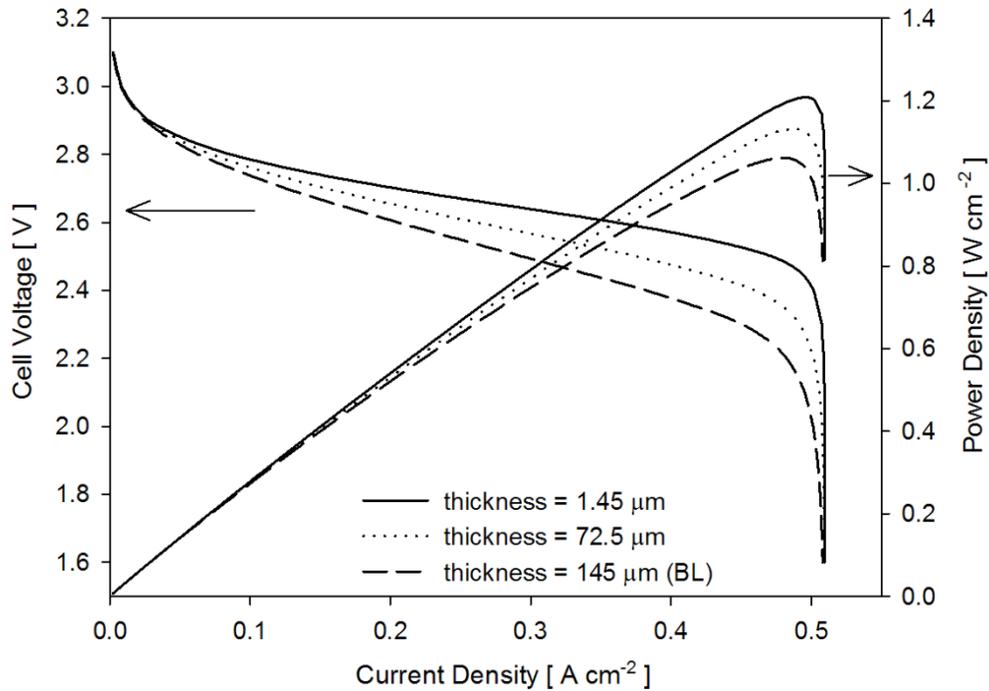


Figure 3.11. Polarization and power density curves for membrane thicknesses of 1.45, 72.5, and 145 μm . All other parameters are the same as in the baseline case, which is labeled “BL”. The thickness of fully hydrated Nafion 115 in the Na^+ form is 145 μm [67].

3.5 Effects of Fuel Concentration and Flow Rate

Changes to fuel concentration and flow rate affect cell performance by changing concentrations near the electrodes, thereby changing the concentration overpotentials. Higher inlet BH_4^- concentration provides a larger gradient to drive BH_4^- from the bulk to the anode, which leads to higher fuel concentration near the anode. Higher flow rate improves convection transport of reactants to the anode and products from the anode. Both scenarios increase power density by decreasing the concentration overpotential, increasing current density, or both.

A plot of power density vs. cell voltage (Figure 3.12) for a range of reasonable BH_4^- inlet concentrations between 0.1 and 0.5 M shows that peak power density is approximately proportional to the BH_4^- inlet concentration. The rate of peak power density increase with concentration slows at higher concentrations (Figure 3.12 inset) because other losses in the cell (ohmic and concentration overpotentials) play a larger role with increasing current density. Because the inlet BH_4^- concentration affects cell performance by altering the concentration overpotential, the strength of the effect depends on cell current density. At high current density the concentration overpotential is large, and increasing the BH_4^- inlet concentration can increase the power density substantially. The fuel utilization at high current density does not change with inlet BH_4^- concentration because both the rate of BH_4^- flow through the cell and the rate of consumption are proportional to concentration. At low current density the concentration overpotential is already small compared to other losses in the cell, and thus, there is little opportunity for higher BH_4^- inlet concentration to affect performance.

The choice of inlet fuel concentration in a DBFC system depends on whether a recirculation loop is used. Given practical lower bounds on fuel flow rate and channel depth, the single-pass fuel utilization will be low – in the case of this study, no greater than 10%. If a recirculation loop is used, it may be preferable to store the fuel at high concentration near the saturation limit and add it to the recirculation loop slowly to maintain low (~ 0.1 M) concentration in the cell. The low fuel concentration in the cell will yield lower peak power density (as shown in Figure 3.12) but will mitigate losses to hydrolysis (R 1.4).

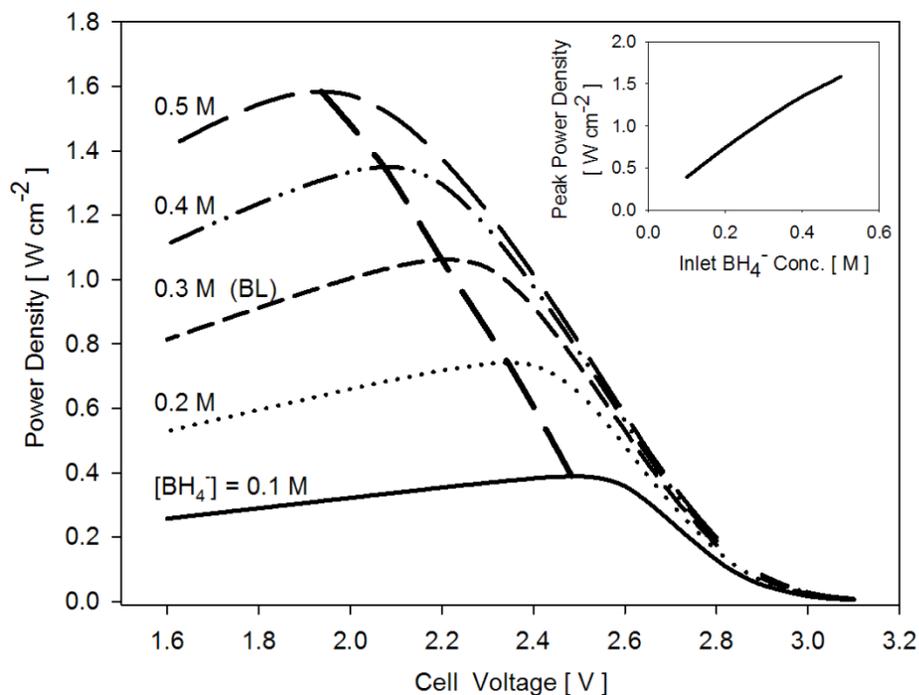


Figure 3.12. Average power density vs. cell potential for fuel inlet concentrations ranging from 0.1 M to 0.5 M. All other parameters are the same as in the baseline case, labeled “BL”. The bold dashed curve is drawn through the maximum power density at each concentration.

Power density is plotted vs. cell voltage for three fuel flow rates in Figure 3.13. The power density increases with flow rate at all current densities as the higher rates of transport change concentrations near the anode. The pressure drop in the channel increases in proportion to the flow rate, but fuel utilization falls as indicated in Table 3-5. Increasing the fuel flow rate by a factor of four increases the pressure drop (pumping losses) by a factor of four, but only increases the peak power by 48%.

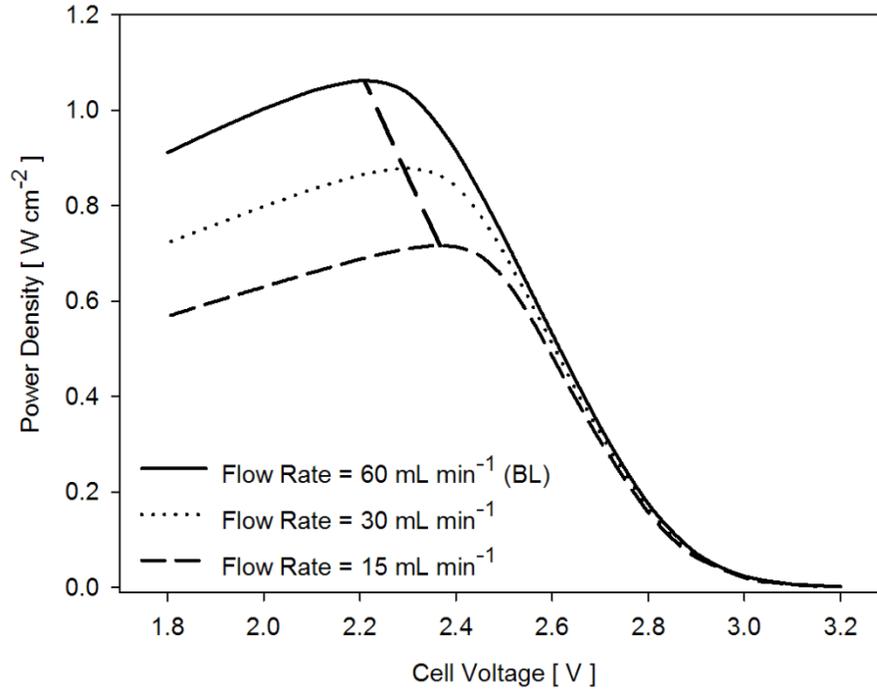


Figure 3.13. Power density vs. cell voltage curves for fuel solution flow rates of 15, 30 and 60 mL·min⁻¹. All other parameters are the same as in the baseline case, labeled “BL”. The bold dashed curve is drawn through the maximum power density at each flow rate.

Table 3-5. Performance metrics with respect to fuel flow rate with all other parameters at the baseline case

Parameter	Fuel Flow Rate [mL·min ⁻¹]		
	15	30	60
Peak power density [W·cm ⁻²]	0.72	0.88	1.06
Cell voltage @ peak power density [V]	2.37	2.29	2.21
Fuel utilization @ peak power density [%]	10.5	6.60	4.14
Power density @ 2.5 V [W·cm ⁻²]	0.65	0.70	0.73
Channel pressure drop [Pa]	51.7	104	210

3.6 *Insights into DBFC Design from the Ideal Case Analysis*

The results in sections 3.2 to 3.5 can guide DBFC design by showing how performance varies with operating conditions and cell design parameters. For conditions in which cell performance is controlled by transport, the assumptions

regarding reaction kinetics and OCV are reasonable, and the model can be used to compare specific design options. For example, Figure 3.12 can be used to choose an inlet BH_4^- concentration by drawing a line through the peak power density of each curve. The dashed line in Figure 3.12 demarcates the trade space maximizing power density as a function of BH_4^- inlet concentration and cell voltage. The region to the right of the dashed line describes designs or operating points which trade power density for higher voltage efficiency. Since the energy density of a system is proportional to fuel concentration, voltage efficiency and fuel utilization, moving to the right in Figure 3.12 can substantially increase the system energy density. The maximum power density achievable given an energy density requirement (or vice versa) can be estimated from Figure 3.12.

In general, shallower channels increase system-level power and energy density due to decreased the stack volume. When the additional volume is used to add active area to the stack, then it can produce the same power at lower current density and operate more efficiently (higher voltage per cell). These trends are seen in the DBFC model results. For example, a stack with channels 1.00 mm deep operating at 2.21 V per cell (point A in Figure 3.9) would have the same volumetric power density as a stack with twice as many 0.50 mm deep channels operating at 2.64 V per cell (point B in Figure 3.9), but the stack with shallower channels would have 36% less voltage loss. Channel fuel utilization is lower in the shallow channel case (3.2% vs. 4.2%) because of the lower current density, and a complete system tradeoff study must include the recirculation fraction and fuel storage density to find an optimum channel depth.

Another performance tradeoff is the increase in peak power vs. parasitic power consumed by the fuel and oxidizer pumps. Table 3-5 shows that pressure drop through the fuel channel is directly proportional to the fuel flow rate, as expected for laminar incompressible flow through a straight channel. The peak power is also proportional to the fuel flow rate, but increases at a rate of less than 8 mW cm^{-2} per mL min^{-1} increase in fuel flow rate. Thus, there may be a design point beyond which the additional power from faster flow is not worth the increase in pressure drop and loss of fuel utilization.

The low channel fuel utilization in all cases in this study bolsters experimental observations that any practical DBFC will require recirculation of the reactants to achieve high (>90%) overall reactant utilization. Near irreversibility of the BH_4^- oxidation reaction makes the anode reaction rate insensitive to BO_2^- concentration; reaction rates varied by less than 0.3% in simulations with BO_2^- inlet concentrations between 10^{-6} M and 0.3 M . The insensitivity to BO_2^- concentration makes extrapolating from once-through simulation results to a recirculating system straightforward because the only parameter related to both cell performance and the recirculation fraction is the BH_4^- concentration at the inlet. A simple recirculation model was constructed which interpolates among the results from the DBFC model with varying inlet concentration of BH_4^- . The recirculation model was used to investigate the relationships between fuel recirculation volume fraction ζ_f , fuel utilization η_{fu} and power density p . Figure 3.14 shows results for one case in which the NaBH_4 fuel is stored at 0.5 M , the cell voltage is 2.5 V and fuel solution is added upstream of the channel at the same volumetric rate waste is rejected downstream of

the channel. Figure 3.14 shows that the fuel utilization improves by a factor of more than 5 over a once-through approach when 99% of the fuel flow is recirculated, at the cost of a 50% decrease in power density due to the lower inlet BH_4^- concentration.

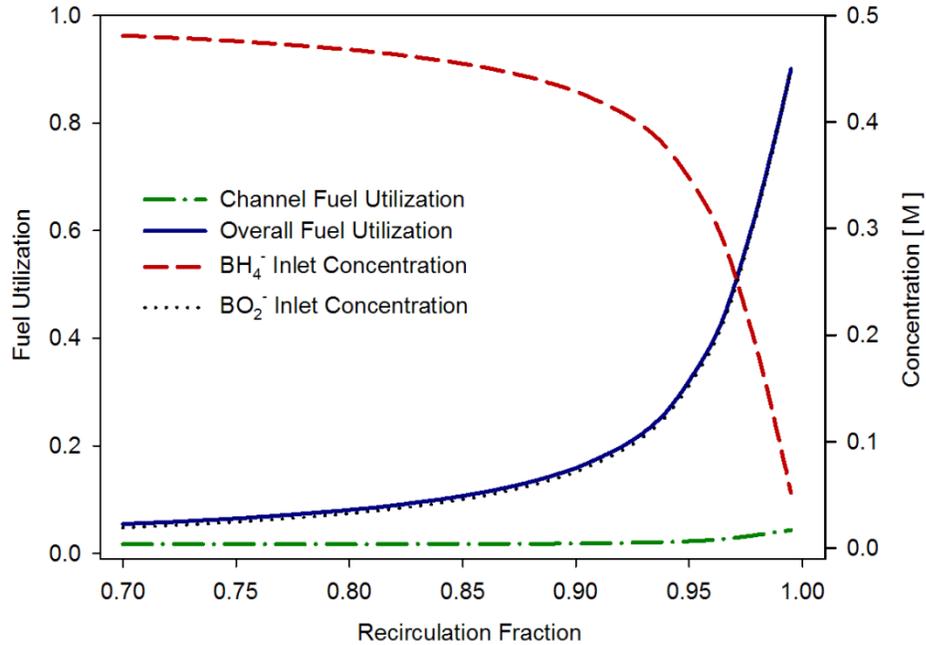


Figure 3.14. Channel (single-pass) fuel utilization, overall fuel utilization, BH_4^- and BO_2^- concentrations at the inlet, all as functions of the fuel recirculation volume fraction. The concentration of BH_4^- in the fuel added to the recirculation loop was assumed to be 0.5 M. Aside from the inlet concentrations, all parameters are the same as in the baseline case.

Chapter 4: Single-Cell DBFC Experiments

4.1 *Goals and Approach*

Models can be used (as in Chapter 3) without calibration or validation when the goal is to explore ideal performance. To make realistic performance predictions, however, the model must be calibrated to measurements of a well characterized system. While many DBFC experiments have been reported, rarely have the systems been described with sufficient fidelity for a reasonable comparison to model results. Furthermore, many experimental DBFC geometries include poorly characterized phases (such as porous catalyst layers) that complicate the process of relating trends in the model results to the measurements.

For this study, a custom DBFC was designed, fabricated and then used to generate polarization curves for comparison to the model. The DBFC has the simple, straightforward geometry of Figure 1.1 with an Au anode and Pd:Ir cathode. Both catalyst layers were characterized electrochemically (cyclic voltammograms), geometrically (optical profilometry) and optically (optical microscopy). The cell was operated with once-through (not recirculated) flows so that the inlet concentrations were known. The flows were driven by a combination of N₂ overpressure and gravity to ensure consistent flow, as opposed to the pulsating flow from peristaltic pumps commonly used with DBFCs. The anode and cathode potentials were measured with respect to reference electrodes to provide insight into the electrochemical environment inside the cell.

4.2 Development of a Single-Cell DBFC and Test Stand

4.2.1 Cell Hardware Design and Fabrication

The cell consists of two flat graphite electrodes separated by flow channel masks and a Nafion 117 membrane (see Figure 4.1). The flow channel masks are 0.5 mm thick sheets of PTFE with oblong 5 mm wide slots cut in them. Each end of the slot overlaps a reactant port, so that when the cell is assembled, the reactants flow through the channels over the graphite plates. The regions of the graphite plates exposed to the flow channels each have an electro-deposited catalyst layer. Each graphite electrode is in contact with a gold-coated brass current collector. Holes through the entire assembly admit compression bolts (see Figure 4.2) for sealing the cell. A common issue in PEMFC fabrication is establishing the correct compressive force to ensure good electrical contact among the various phases without crushing porous media (diffusion and catalyst layers). The performance of this DBFC is insensitive to compressive force as long as there is sufficient force to seal the layers, because none of the components are porous and all of the relevant electrical conductors are in intimate contact. The channel masks have large enough area that the channel dimensions are stable despite large compressive forces. The low pressure of reactants (marginally above ambient pressure) made sealing the cell straightforward. The compliance and hydrophobicity of the PTFE channel masks and the smoothness of the graphite plates together discouraged fluid entry between the layers. The large channel mask areas, when compared to the channel areas, ensured dimensional stability in the direction perpendicular to the graphite plates so that the channel depth was dictated by the mask thickness. The channels were made as wide as possible

while still supporting the membrane in the center of the gap between the graphite plates.

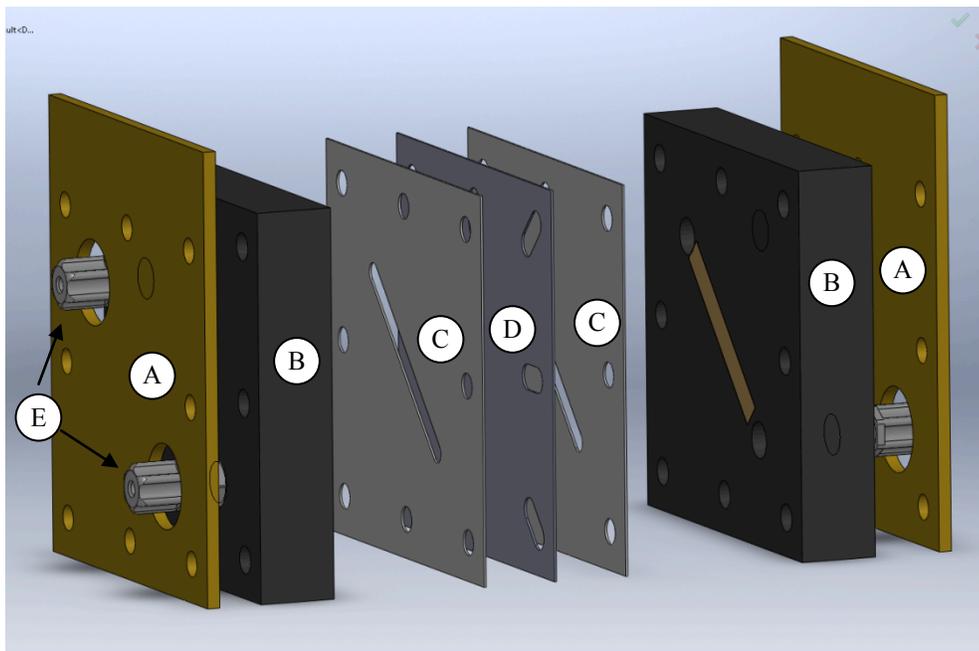


Figure 4.1. Solid model of the DBFC used for calibration experiments. (A) Au-coated brass current collectors, (B) graphite electrodes, (C) PTFE channel masks, (D) Nafion 117 membrane, (E) PVDF reactant inlet and outlet.

Components were selected to resist chemical attack by the corrosive reactant solutions. The Nafion membrane, PTFE channel masks and graphite plates are all resistant to chemical attack. PVDF compression fittings were used for the reactant ports, and PTFE tubing was used to deliver the reactants and remove the effluent streams. A PVDF “T” fitting was added to each reactant line at the cell inlet to admit a Ag/AgCl reference electrode.

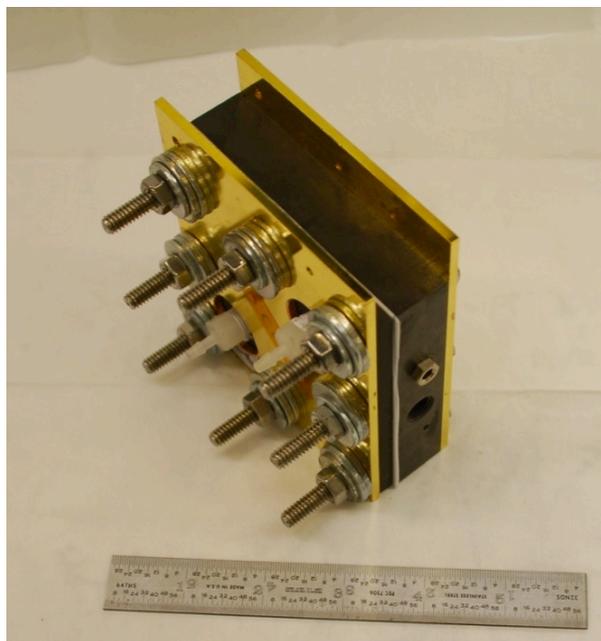


Figure 4.2. Photograph of the DBFC used for calibration experiments.

4.2.2 Electro-deposition of Electrodes

The process used to electro-deposit the DBFC electrocatalyst layers was adapted from those described by Miley [85], de Leon [14], Urian [1] and Besette [58]. The electrocatalyst layers for the anode and the cathode were deposited separately; in each case the cell, was assembled with the electrode to receive the catalyst layer (working electrode, or WE) and a graphite counter electrode (CE). The membrane was omitted so that the plating solution would be in contact with both electrodes. The channel masks were modified to have wider (7 mm) channels for plating to eliminate registration errors when the cell was assembled for experiments. The channel mask for the WE was further modified to mask the region near the reactant ports and provide a sharply defined end to the catalyst layer (see Figure 4.3).

The graphite plates were prepared by cleaning and then modifying the surface to accept the electro-deposited metals. While no oil was used when machining the plates, they may have been contaminated by trace oil on the cutting tools and/or mill. The plates were rinsed with methanol and acetone to remove traces of machine oil, and then dried and rinsed with 18 M Ω water. The surfaces to be plated were lightly sanded with coarse (P80 grit) Al₂O₃ sandpaper and then polished with fine (P600 grit) SiC sandpaper. Graphite powder left behind by sanding would have prevented good adhesion between the electro-deposited metals and the surface, so it was removed by sonication. The plates were placed in plastic zip-lock bags with 18M Ω water and submerged in an Ultrasonic Power Corp. model 5300/50-26-459 sonication bath for 20 min at 200 power. The water surrounding the plates became opaque during sonication, and pre- and post-sonication micrographs showed that sonication removed graphite dust from pores and scratches.

Plating solutions were circulated through the cell from a reservoir (beaker) by a peristaltic pump (see Figure 4.4) through 1/8 inch inside diameter flexible PVC tubing. An Autolab PGSTAT30 potentiostat/galvanostat drove current through the cell while monitoring the WE potential with respect to a Ag/AgCl RE in the beaker. The current was pulsed repeatedly; 5 mA·cm⁻² for 2.5 s and 0 mA·cm⁻² for 2 s. The plating solution residence time in the cell was ~1 s, so this procedure allowed complete replacement of solution in the cell between pulses. The number of pulses (total charge transfer) was chosen to give ~20 mg·cm⁻² coverage (assuming 1:1 ratio for the Pd:Ir). The inlet and outlet ports on the cell were reversed half-way through

the plating process to mitigate layer thickness variation from inlet to outlet. Plating took place at room temperature ($\sim 23^{\circ}\text{C}$).

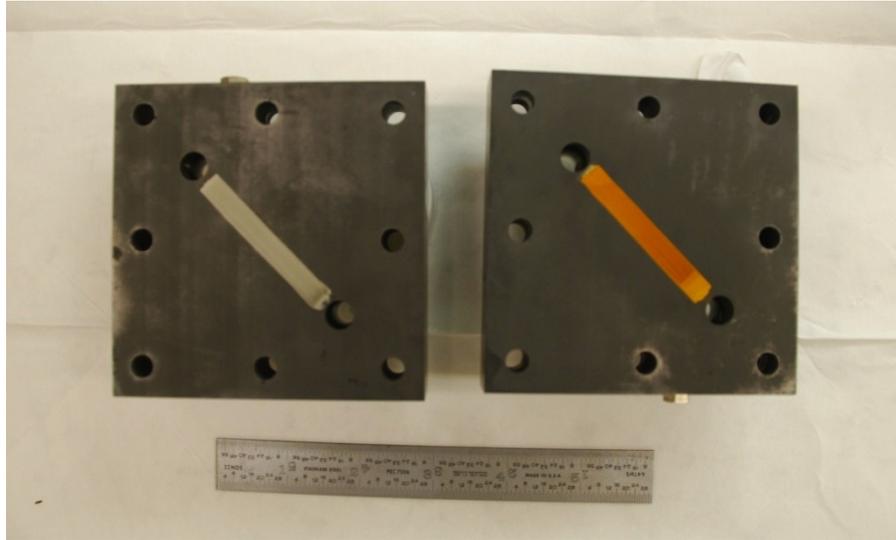


Figure 4.3. Cathode (left) with Pd:Ir catalyst layer and anode (right) with Au catalyst layer.

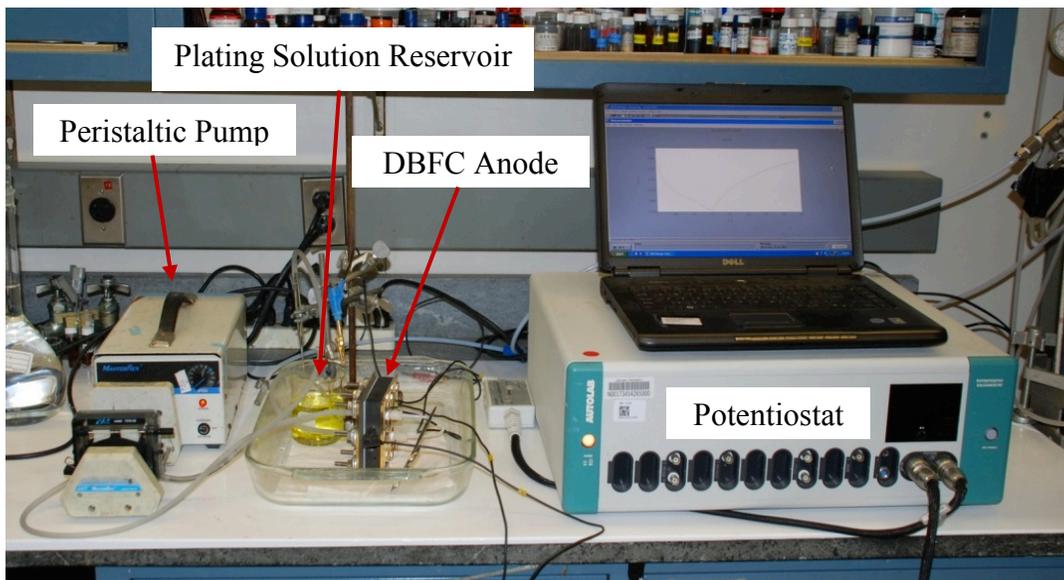


Figure 4.4. Photograph of the setup for plating the DBFC anode.

The anode plating solution consisted of 2 mM NaAuCl₄ and 200 mM NaCl in 18 MΩ water. The cathode plating solution consisted of 2 mM PdCl₂, 2 mM Na₃IrCl₆, 200 mM KCl and 0.1 M HCl in 18 MΩ water. Each solution was prepared using Alfa-Aesar Premion reagent grade salts. The initial volumes of plating solution in the reservoir were 100 mL, which provided sufficient salt for the desired deposition thickness, plus substantial margin. Some salt was evident in the plating solutions after plating due to the color (light yellow for the anode solution and brown for the cathode solution).

4.2.3 Membrane Preparation

The Nafion 117 membrane was prepared for use by cleaning and then conversion to the Na⁺ form. To clean the membrane, it was boiled in 3% H₂O₂ for 1 hr and then rinsed with 18 MΩ water. To convert to the Na⁺ form, the membrane was boiled in 1% NaOH for 1 hr and then stored in the NaOH solution. The membrane was rinsed with 18 MΩ water prior to cell assembly.

4.2.4 Test Stand Setup

A test stand (shown in Figure 4.5) was built to control the flow of reactants through the cell. Reactants were stored in 5 L capacity HDPE carboys located on a shelf above the cell, and flowed to the cell through HDPE ball valves and 1/16 inch inside diameter PTFE tubing. The carboy caps were modified to accept two PVDF compression fittings and a dip tube. The flows were driven by a combination of gravity and 2.5 psig N₂ overpressure. The N₂ overpressure was controlled by an AirTrol R-800-01 pressure regulator and monitored by a MILJOCO 0-5 psig analog pressure gauge with ±5% full scale accuracy.

Additional 1/16 inch inside diameter PTFE tubing carried the effluent streams from the cell to a flow control board located below the cell. On the flow control board, fixtures permitted 0.02 inch inside diameter PTFE tubing to be spliced into the flow paths to add pressure drop and slow the flows. Droplet formation can affect the flow rate, and so inlets to the flow control board were on the bottom to ensure the outlets remained submerged. Calibration curves were generated by measuring the reactant flow rates with various lengths of tubing in the flow control board and N₂ overpressures in the carboys.

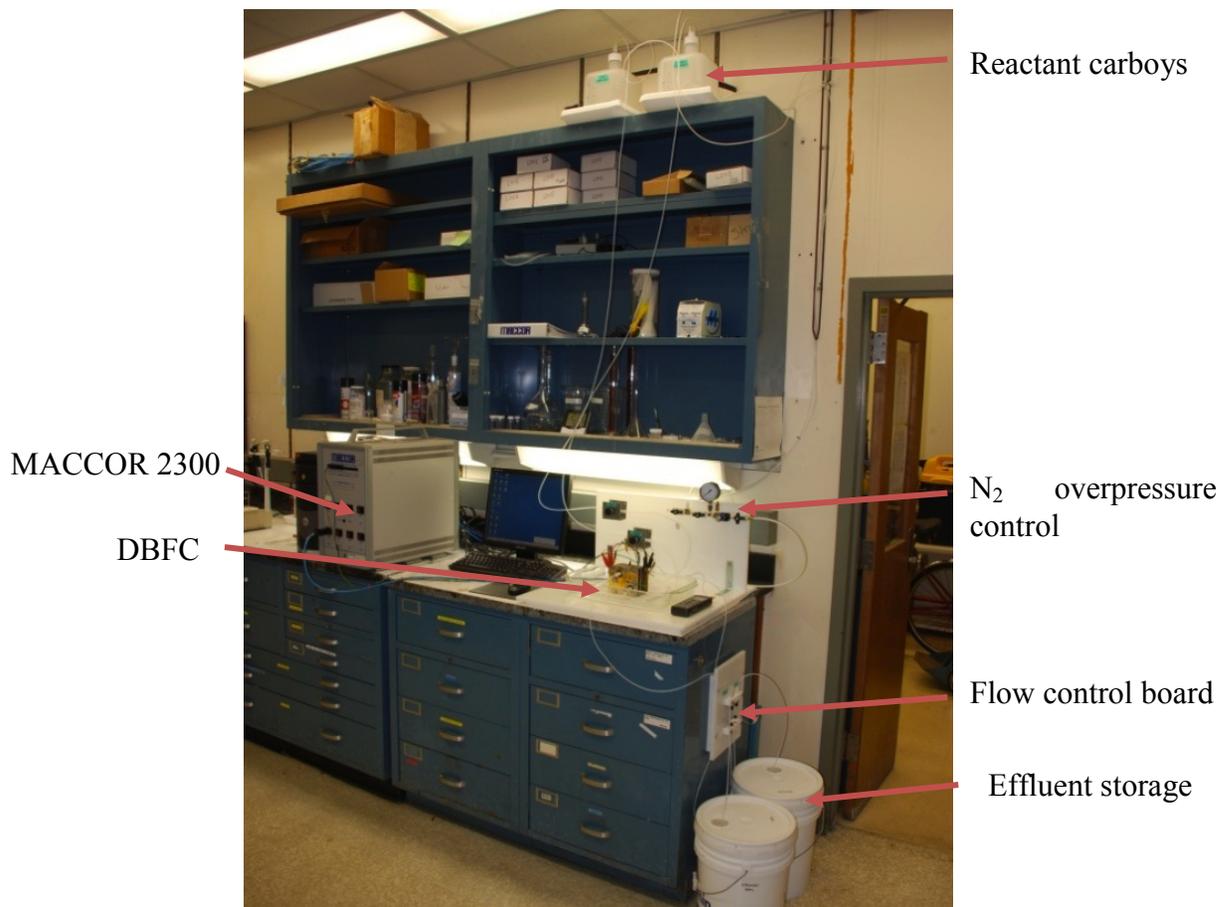


Figure 4.5. Photograph of the test stand with DBFC.

A MACCOR Model 2300 battery test system was used to set the cell voltage and measure the corresponding current. The MACCOR also recorded the potential of each cell electrode with respect to the appropriate reference electrode. The MACCOR electric potential measurement subsystem was calibrated against an Autolab PGSTAT30 prior to the experiments. The cell temperature was measured by a type K thermocouple affixed to the anode current collector plate by Kapton tape, and read by an Omega Engineering HH303 thermocouple reader.

4.3 Characterization of Electrodes

4.3.1 Optical Microscopy and Profilometry

The coverage density of each electrode was investigated by optical microscopy. Figure 4.6 shows the graphite anode plate with the boundary between Au-coated and bare regions. Two conclusions can be drawn from Figure 4.6. First, the plated Au uniformly covers the anode surface, and second, the PTFE channel masks seal against the graphite plates well enough to limit the electrochemically active area to the desired channel dimensions.

The electrode topologies were investigated using optical profilometry. Each electrode was scanned by a STIL model CHR 450 optical profilometer with a 1.5 x 1.5 μm grid. An electrode plated without first sanding the surface (Figure 4.7) showed periodic height variations of $\pm 2 \mu\text{m}$ which were artifacts of the machining process. An anode electrode plated after sanding and sonication showed more uniform height (see Figure 4.8) with $8 \pm 2 \mu\text{m}$ electrode thickness (see Figure 4.9).

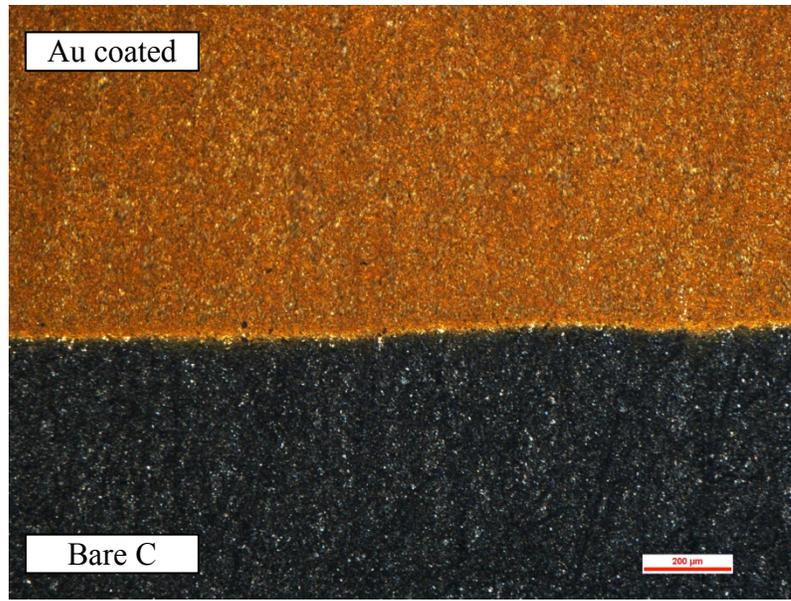


Figure 4.6. Micrograph showing the boundary of the Au-coated and uncoated regions of the anode. Scale bar is 200 μm .

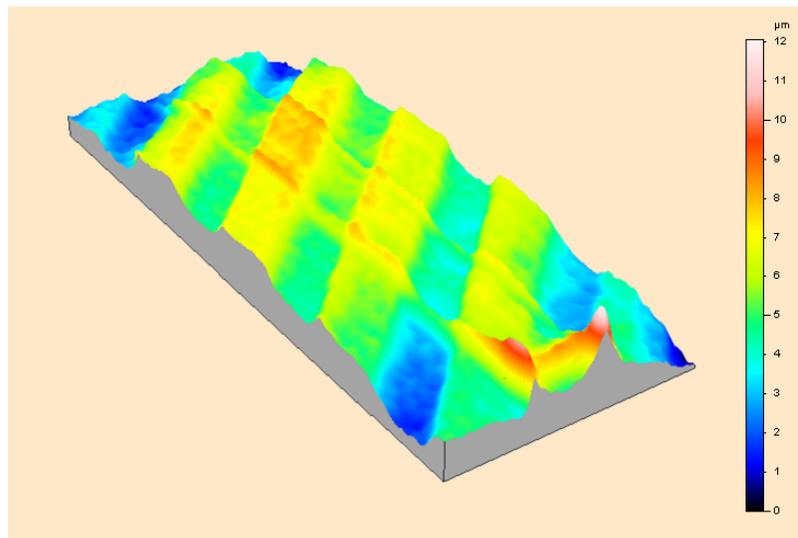


Figure 4.7. Electrode plated without first sanding the surface, which was dominated by artifacts of the plate machining process.

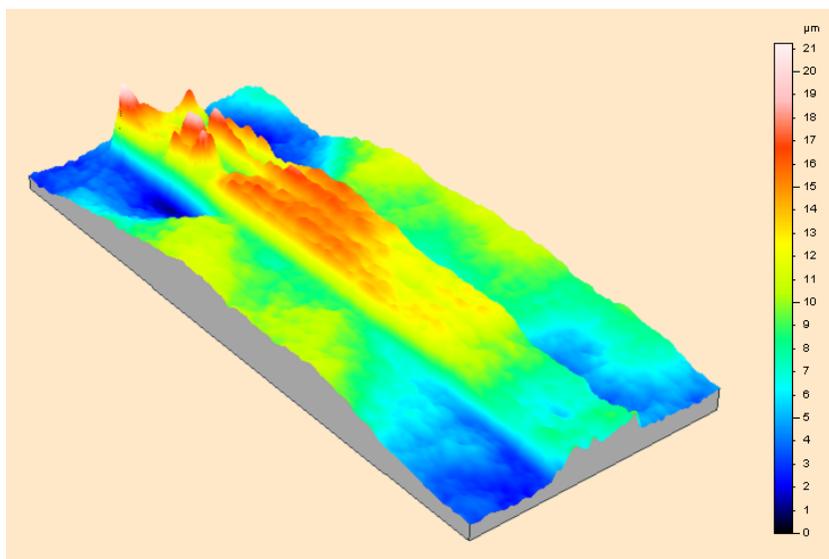


Figure 4.8. Anode with Au strip electro-deposited after sanding the graphite plate. Peaks in Au height correspond to the inlet during the first phase of electro-deposition.

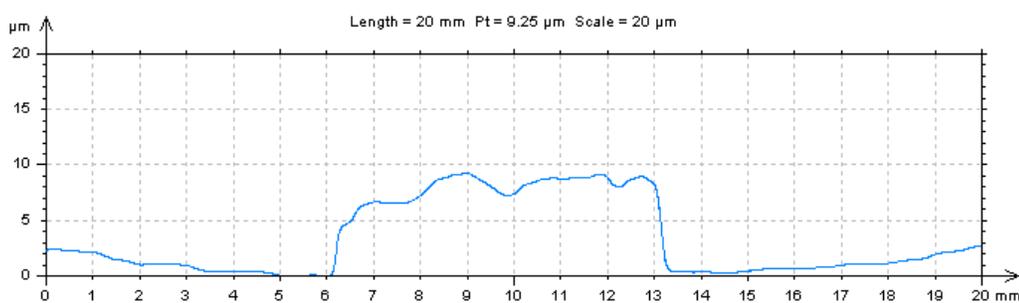


Figure 4.9. Height profile across the middle (2.5 cm from each end) of the anode Au strip.

4.3.2 SEM and EDS

The deposition of Au was straightforward in that only one species was involved. The cathode catalyst layer, however, was a mixture of Pd and Ir whose atomic ratio was determined by the relative rates at which the two species plated out of solution. The different mobilities charges (Pd^{2+} and Ir^{3+}) could produce substantially different deposition rates and a Pd:Ir ratio which differed from the desired 1:1. Furthermore,

the deposition rates could have varied from the inlet to outlet of the cell as the plating solution was depleted. Electron dispersive x-ray spectroscopy (EDS) was used to measure the abundance of Pd and Ir at both ends and the middle of the cathode electrocatalyst strip. The results show a trend of increasing Pd deposition rate from inlet to outlet, with the Pd:Ir ratio varying from 1.1:1.0 near the inlet to 1.4:1.0 near the outlet. The abundance of C varies from 13.3% near the inlet to 15.4% near the outlet, suggesting the plating solution was becoming depleted and/or a concentration boundary layer developed during the deposition process. Small quantities (< 2.5%) of Cl and Na were also evident, and were most likely traces from the plating solution. A study by Zhang et al. [86] indicated that Ir exists in this solution (at low pH) primarily as $[\text{IrCl}_6]^{-2}$, so Cl^- may have deposited independently, or as part of the Ir deposition process as $[\text{IrCl}_6]^{-2}$ reached the surface.

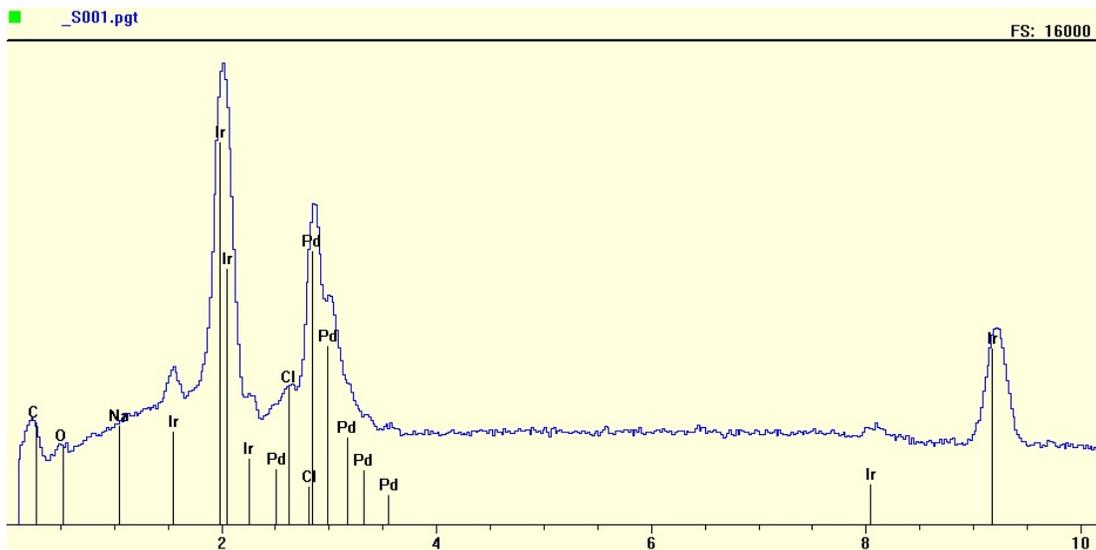


Figure 4.10. Example EDS spectrum from end “A” of the cathode electrocatalyst strip, showing relative abundances of each species.

Table 4-1. Species abundances (%) in the cathode electrocatalyst strip, measured by EDS.

Species	Location		
	End "A"	Middle	End "B"
O	4.51	3.59	4.36
C	13.29	14.80	15.38
Na	0.01	0.51	0.39
Ir	38.45	35.36	32.42
Pd	42.16	43.55	45.36
Cl	1.57	2.19	2.09

Scanning electron microscopy (SEM) was employed to investigate the morphology of the two catalyst strips. The Pd:Ir cathode was examined before and after the model calibration experiments, and no obvious change was observed in the surface morphology or graphite plate coverage. An image of the cathode at 100X magnification (Figure 4.11) shows uniform surface coverage with small crevices that may or may not reveal the graphite plate underneath. In either case, the cathode coverage is nearly 100%. An image at 10⁴X magnification shows a surface comprised of many rounded features which presumably grew and merged during the plating process (Figure 4.12). The Au anode exhibited a degree of surface coverage similar to the cathode (approaching 100%), but a different surface morphology with greater texture, as shown in Figure 4.13.

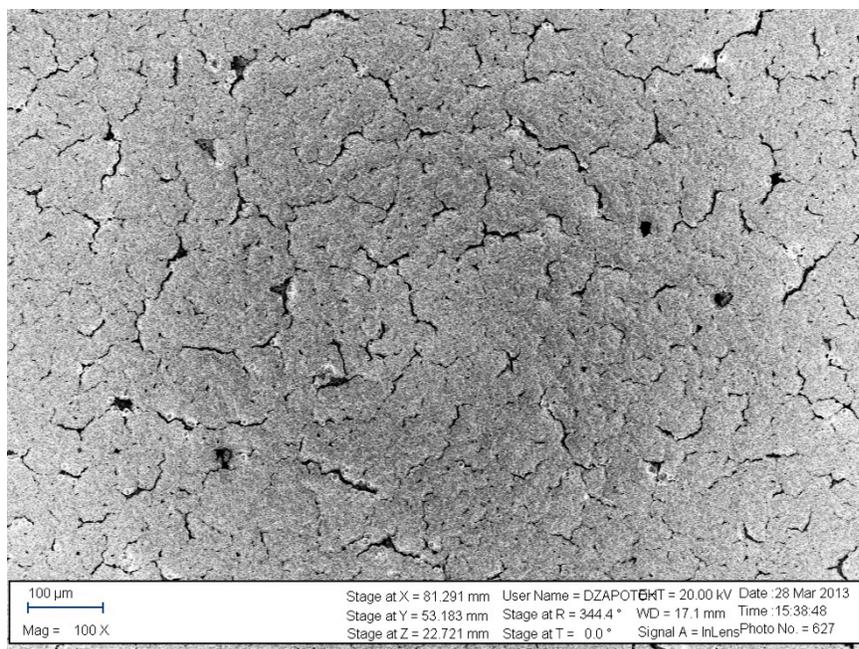


Figure 4.11. SEM image of the Pd:Ir cathode prior to experiments, showing good coverage of the graphite plate.

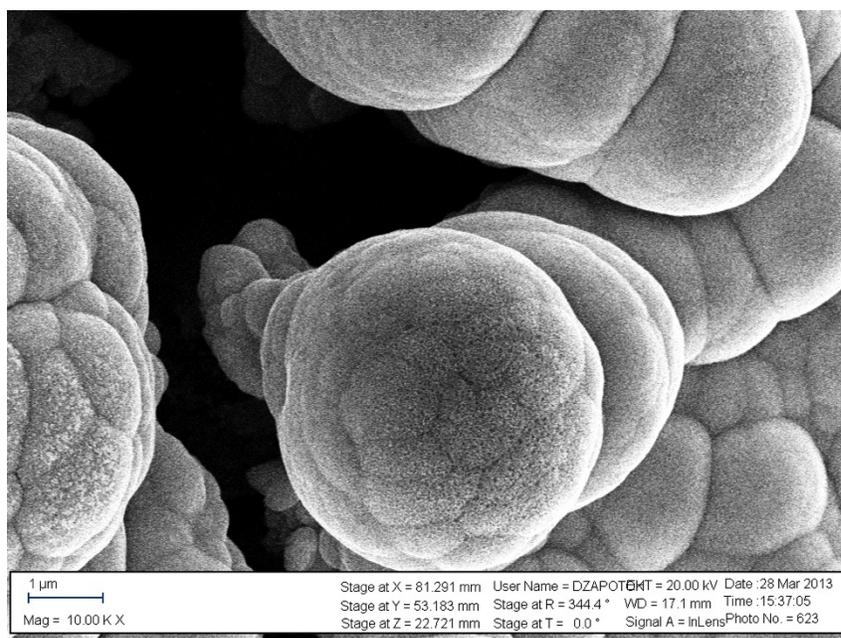


Figure 4.12. SEM image of the Pd:Ir cathode prior to experiments, showing surface morphology consisting of rounded features.

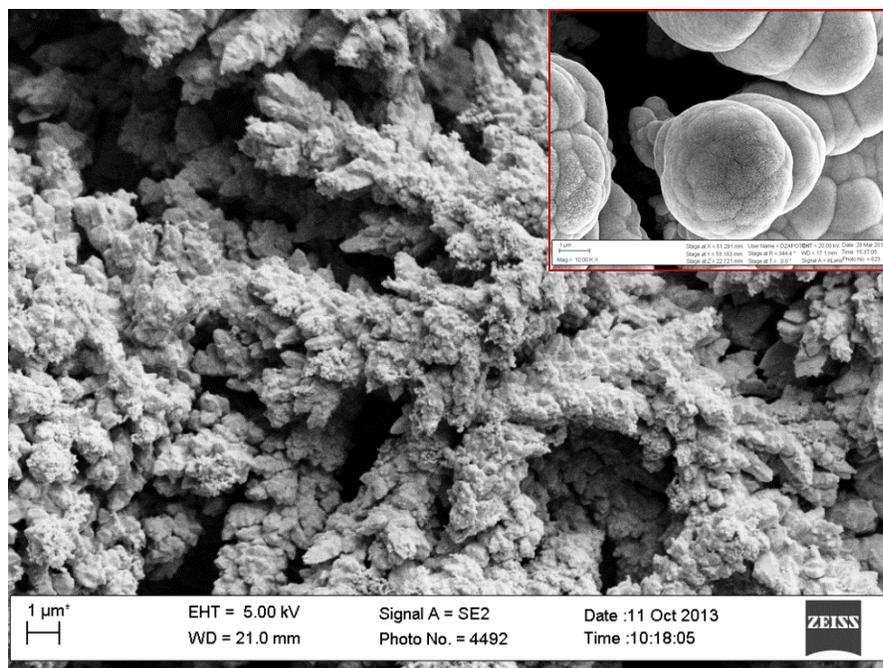


Figure 4.13. SEM image of the Au anode prior to experiments, showing a different surface morphology with more texture than in the Pd:Ir cathode. The cathode image is inset with size adjusted to match the anode scale bar.

4.3.3 Cyclic Voltammograms

The electrochemical characteristics of the two electrodes were evaluated by cyclic voltammetry (CV). A procedure similar to the plating procedure was used to measure each CV. The electrode of interest was assembled in the cell with a blank (i.e. graphite only) CE and no membrane, and 0.5 M H₂SO₄ was pumped through the cell from a reservoir by a peristaltic pump. The H₂SO₄ was de-aerated by bubbling Ar through it while stirring. The pump and stir plate were halted during measurements to minimize electromagnetic interference, but the cell remained connected to the reservoir to ensure ionic conductivity between the cell and Ag/AgCl RE in the reservoir. CVs were generated by an Autolab PGSTAT30 potentiostat/

galvanostat with scan rate of 20 mV s^{-1} , after 10 rapid cleaning scans. The resulting CVs are presented in Figure 4.14.

The Au CV shows the characteristic features found in published Au CVs under similar conditions [87]. Integrating the charge under oxide reduction peak (labeled *a* in Figure 4.14; 0.5 V to 1.1 V) and dividing by the charge density of one oxygen monolayer ($0.42 \text{ mC}\cdot\text{cm}^{-2}$ [88]) yields an electrochemical surface area (ECSA) of 47.1 cm^2 . Dividing the ECSA by the 2.5 cm^2 geometric surface area yields an anode roughness factor of approximately 18. The DBFC cathode roughness factor was not calculated from the area under the oxide peak labeled *b*, as it was for the anode, because the charge density of an oxide monolayer on this alloy is not known.

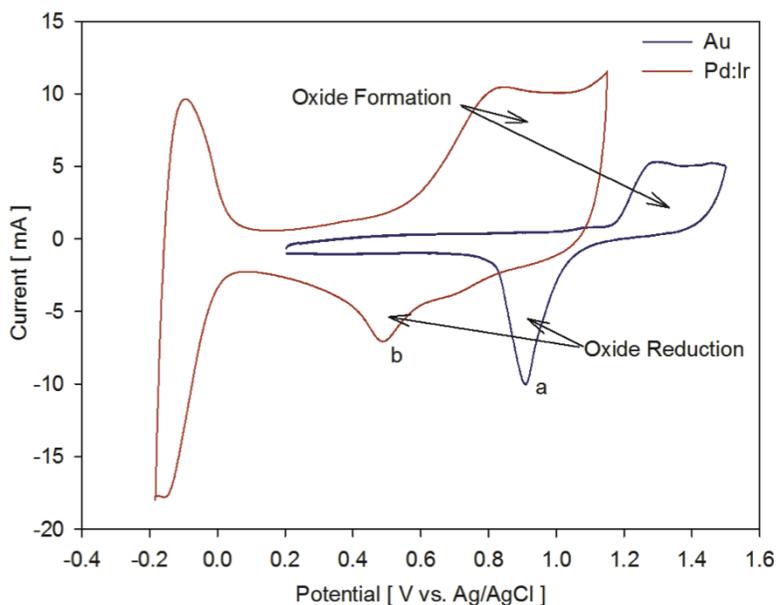


Figure 4.14. Cyclic voltammograms for the Au anode electrocatalyst and Pd:Ir cathode electrocatalyst. Measured in $0.5 \text{ M H}_2\text{SO}_4$ with a scan rate of $20 \text{ mV}\cdot\text{s}^{-1}$.

There are few examples of Pd:Ir CVs in the literature; the most similar CVs available were reported for various ratios of Pt:Ir in 0.1 M HClO_4 by Chen and Chen

[89]. The Pd:Ir CV in Figure 4.14 most resembles the CVs in [89] for Pt:Ir ratios of 66:43 and 1:0. Kjeang [90] reported a CV of a planar Pd electrode deposited on a graphite plate in the course of experiments with a microfluidic formic acid fuel cell. Unfortunately, the Kjeang CV is nearly featureless, perhaps due to poor deaeration of the electrolyte.

4.4 Test Preparation and Procedures

4.4.1 Reactant Preparation

All glassware used for reactant preparation and wetted parts in the test stand (carboys, valves, etc.) were cleaned with NaHCO_3 and 98% H_2SO_4 , and then rinsed thoroughly with 18 M Ω water. The fuel solution was prepared by adding solid NaOH (Fisher Scientific) to 18 M Ω water and then adding the desired amount of solid NaBH_4 (Alfa Aesar). The oxidizer solution was prepared by diluting 98% H_2SO_4 (Fisher Scientific) with 18 M Ω water and then adding the desired amount of 30% wt H_2O_2 solution (Fisher Scientific). The base and acid were each prepared first so that the NaBH_4 would not hydrolyze and the H_2O_2 would not decompose when added. Both reactant solutions were prepared less than 30 min prior to each experiment to minimize changes in BH_4^- and H_2O_2 concentration due to hydrolysis and decomposition. The reactants were permitted to achieve thermal equilibrium with the laboratory environment prior to each test. Neither reactant solution was deaerated prior to the experiments, although the N_2 overpressure would have lowered the dissolved O_2 concentrations slightly. Au has little activity for O_2 reduction, so dissolved O_2 should not have affected the anode behavior. Pd:Ir, on the other hand,

has high activity for O₂ reduction and dissolved O₂ may have provided an additional source of oxidizer. The O₂ concentration in pure water in equilibrium with 1 atm air at 25°C is 2.67×10^{-4} M [91], or 150 times smaller than the 40 mM H₂O₂ concentration in most of the experiments.

4.4.2 Measurement and Test Procedures

For each experiment, the carboys were pressurized to 2.5 psig and then the reactant valves were opened. The cell was permitted to equilibrate at open circuit for 5 min, and then the MACCOR test script was started. The test script began with a 5 min hold at open circuit and then generated a polarization curve by stepping through cell potentials from 0.3 V to open circuit. Each subsequent cell potential was held for 2.5 min. The short hold periods (compared to common PEMFC test procedures) were found to be sufficient because the cell rapidly approached steady state, as judged by observing the cell current. The period at 0.3 V was longer to ensure the cathode was thoroughly reduced and improve consistency among polarization curve measurements. Each test consisted of three successive polarization curves which were later used to compile an average curve and standard deviation for each point. Current was measured at 1 Hz, and the last 2 min of each hold period were used to compile an average cell current for that step.

The reactant flow rates were measured periodically (twice per experiment) to ensure they were consistent among experiments. The rates were measured by directing the flows into 250 mL graduated cylinders and measuring the time to fill them. At $10 \text{ mL} \cdot \text{min}^{-1}$ (the flow rate for most of the experiments) the time measurement error was insignificant compared to the total time of ~25 min, making

the measurement uncertainty depend entirely on the accuracy of the graduated cylinders. Compiling all flow rate measurements gave values of $10.17 \pm 0.46 \text{ mL} \cdot \text{min}^{-1}$ for the fuel and $10.05 \pm 0.16 \text{ mL} \cdot \text{min}^{-1}$ for the oxidizer.

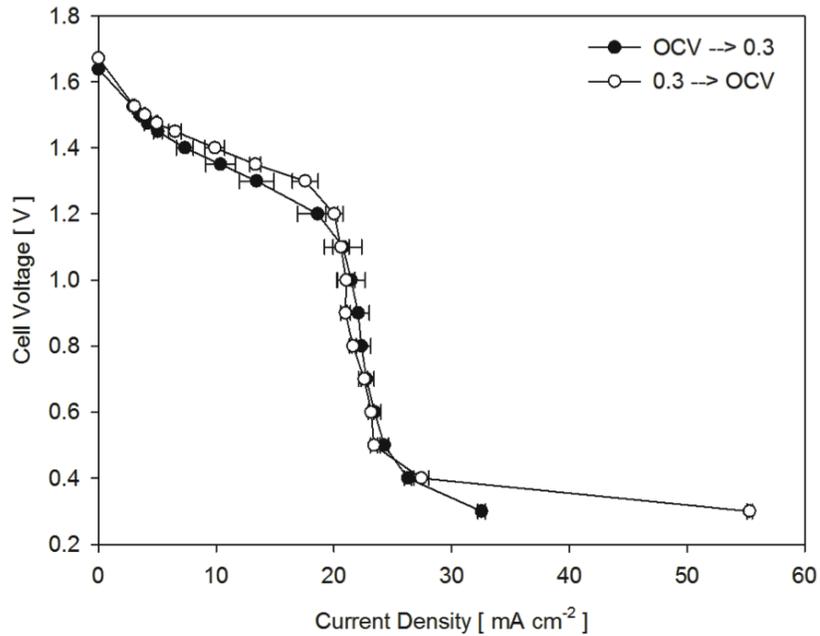


Figure 4.15. Polarization curves measured with 20 mM BH_4^- / 2 M NaOH fuel and 40 mM H_2O_2 / 1 M H_2SO_4 oxidizer. One curve was generated by stepping from open circuit to 0.3 V, and the other by stepping from 0.3 V to open circuit.

Oxidizing conditions such as high electrode potential or exposure to strong oxidizers have been known to oxidize fuel cell electrocatalysts, which decreases their catalytic activity. The Au anode catalyst remained in the reduced state in the strongly reducing anode fuel flow, but the Pd:Ir cathode was exposed to H_2O_2 (a strong oxidizer) and expected to take on high electric potential. To evaluate the influence of cathode catalyst oxidation on the results, two polarization curves were measured, one stepping from open circuit down to 0.3 V and one stepping from 0.3 V up to open circuit. The two curves are shown in Figure 4.15, where some minor differences are

apparent. The curve stepping from 0.3 V to open circuit has higher current densities in the ohmic and activation overpotential regions, as expected for a catalyst layer which was more reduced by operating at low electric potential prior to measuring the curve.

Each set of experiments began and ended with a baseline polarization curve to show whether or not the cell state (for example, catalyst oxidation state) was consistent for the intervening experiments. The baseline case was 10 mM NaBH₄ / 2 M NaOH fuel and 40 mM H₂O₂ / 1 M H₂SO₄ oxidizer. 1:1 stoichiometry (assuming the ideal reaction, R 1.1) was chosen for the baseline case so that a change in the activity of either electrocatalyst would appear in the polarization curve; otherwise changes in an electrode could be masked by an excess of reactant.

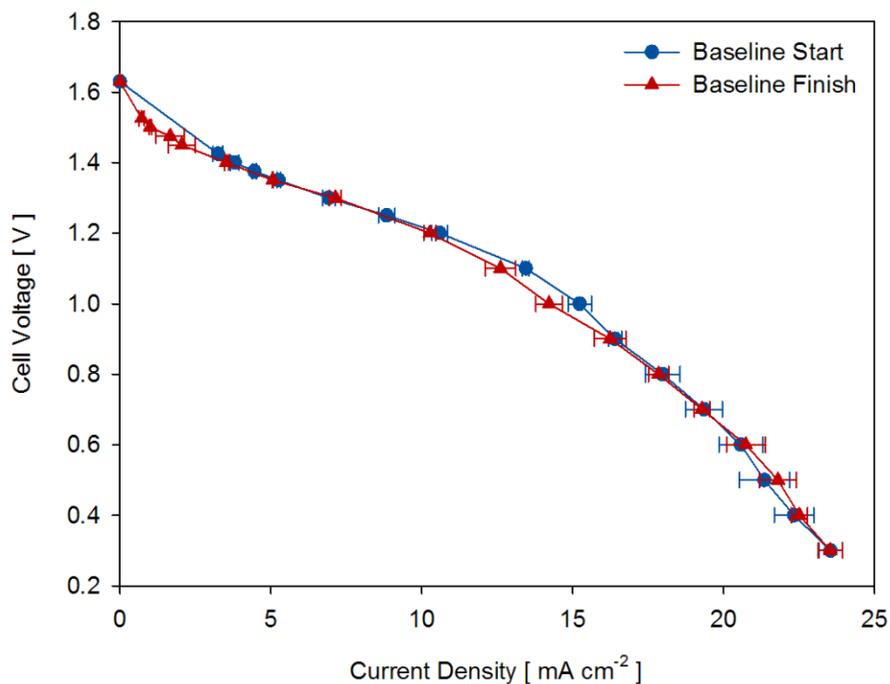


Figure 4.16. Comparison of baseline (10 mM NaBH₄ / 2 M NaOH fuel, 40 mM H₂O₂ / 1 M H₂SO₄ oxidizer) polarization curves, measured before and after experiments.

Figure 4.16 shows baseline polarization curves measured before and after the bulk of experiments discussed in this chapter. They vary little, showing that the cell state was likely consistent throughout the test matrix. The apparent discrepancy in the activation region is the result of adding several points in the “Finish” polarization curve to better resolve the curve in that region.

4.5 *Results and Discussion*

4.5.1 Measured Polarization Curves and Electrode Potentials

Five polarization curves were measured with BH_4^- concentrations ranging from 1 to 20 mM, all in 2 M NaOH. Concentrations in the oxidizer solution were held constant at 40 mM H_2O_2 / 1 M H_2SO_4 . This BH_4^- concentration range was chosen for two reasons. First, BH_4^- oxidation is the least understood reaction occurring in the cell, with both the mechanism and rate in doubt. Varying BH_4^- concentration was expected to probe the anode reaction(s) and show how the relative rates change, because BH_4^- is likely involved in the rate-determining step. If for example, the anode mechanism consists of R 1.2 and R 1.4, then changing BH_4^- concentration should change the relative rates of those reactions and the current density. Variation in polarization curves with operating conditions was expected to permit fitting of rate parameters to the measurements. The second reason for choosing this range of BH_4^- concentrations was to vary the transport limiting species. 1 mM, 2.5 mM and 5 mM BH_4^- give fuel limited stoichiometries. 10 mM BH_4^- gives 1:1 stoichiometry, and 20 mM BH_4^- is oxidizer limited. Varying the limiting reactant was expected to provide

insight into the transport behavior of these species and test the accuracy of transport rate prediction by the model.

The five measured polarization curves are shown in Figure 4.17. All five curves exhibit clear activation overpotential regions between OCV and 1.4 V. As expected, the ohmic overpotential region is most pronounced in the 20 mM BH_4^- curve, which has the highest current density.

Several differences between the measured polarization curves and those predicted by the ideal DBFC analysis are apparent. First, OCV is depressed in the measured curves by ~ 1.4 V. The ideal DBFC analysis presumed that no electrode reactions occur at OCV, so that reactant concentrations near the electrodes were equal to the bulk concentrations. The lower measured OCVs can be explained by competing reactions consuming reactants near the electrodes, which lowered the local concentrations and depressed electrode potentials as predicted by the Nernst equation. Furthermore, the real reaction rate constants were likely smaller than the fast values assumed for the ideal analysis. Smaller rate constants lead to larger activation overpotentials, because electrode potentials must shift further from equilibrium to achieve even small net current density. Large activation overpotentials near OCV can appear to be shifts in OCV as small current densities are masked (appear to be zero current density) by effects such as leakage currents and membrane crossover.

A second difference between the measured polarization curves and those predicted by the ideal analysis is the onset of the transport limit. In the ideal analysis, current density suddenly ceased to increase with decreasing cell voltage at the transport limit. The measured 1 mM and 2.5 mM BH_4^- concentration curves behave

this way, but the 5 mM, 10 mM and 20 mM curves end differently. The 5 mM and 10 mM BH_4^- concentration curves approach the transport limit gradually with current density rising less quickly as the cell voltage is decreased. The 20 mM BH_4^- is similar, but in the 0.4-0.6 V range the current density appears to increase more quickly as the cell voltage is decreased. These trends indicate the presence of a process which was omitted in the ideal DBFC analysis.

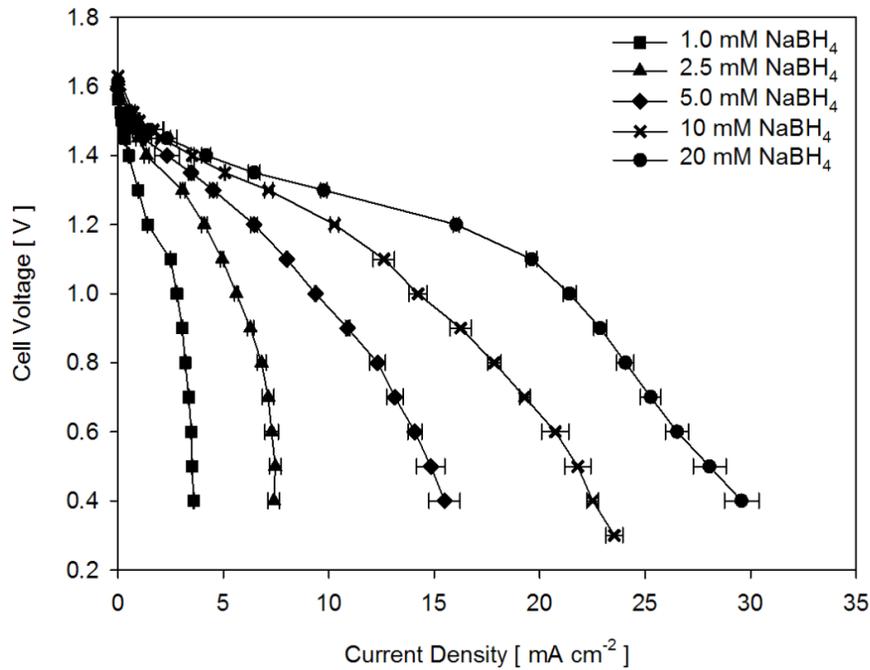


Figure 4.17. Measured polarization curves with varying BH_4^- concentration in 2 M NaOH. In all cases, the oxidizer was 40 mM H_2O_2 / 1 M H_2SO_4 . Fuel and oxidizer flow rates are both $10 \text{ mL} \cdot \text{min}^{-1}$.

A plot of current density vs. BH_4^- concentration for each cell voltage (see Figure 4.18) suggests that the transport limited current density varied linearly with BH_4^- concentration when $C_{\text{BH}_4^-} \leq 5 \text{ mM}$. This is reasonable given that the fuel cell was operating with fuel limited stoichiometry when $C_{\text{BH}_4^-} < 10 \text{ mM}$. The transition from

fuel limited to oxidizer limited operation is not dictated by stoichiometry alone, however. The diffusivity of BH_4^- ($2.42 \times 10^{-9} \text{ m}^2 \cdot \text{s}^{-1}$) is greater than that of H_2O_2 ($1.49 \times 10^{-9} \text{ m}^2 \cdot \text{s}^{-1}$), and the BH_4^- flux is aided by migration, so the transition is likely to occur at lower BH_4^- concentration than 1:1 stoichiometry. This may explain why the highest current density curves (at 0.4 V) in Figure 4.18 change slope between BH_4^- concentrations of 5 mM and 10 mM.

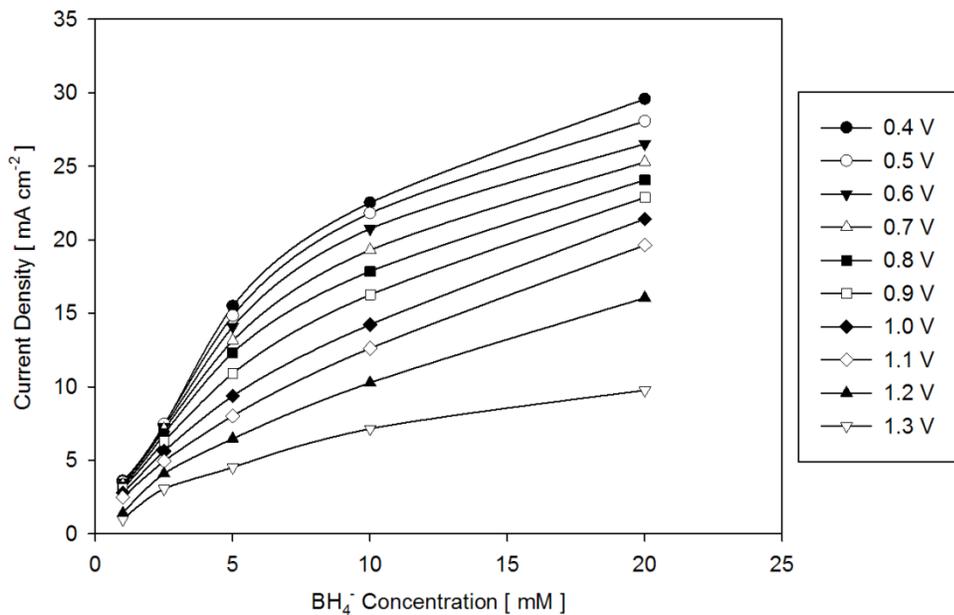


Figure 4.18. Plots of measured current density vs. BH_4^- concentration for the specified cell voltages.

There are several possible explanations for the gradual decline in current density in the higher BH_4^- concentration curves. For example, the shift in concentrations at one or both electrode interfaces as the current density increases could lead to a change in the relative rates for reactions occurring there. The shifts in relative rates may favor charge transfer reactions (for example, favoring reaction R 1.2 over reaction R 1.4) and postpone the appearance of a hard transport limit to lower cell voltages.

Another possibility is that migration aids transport of the limiting species. In this case, the transport limited region of the polarization curves should have the observed shape. The migration fluxes are proportional to the local electric potential gradient, and lower cell voltage may create a larger electric potential difference across the channel containing the limiting species. The result would be a flux of the limiting species which varies linearly with cell voltage.

Finally, a third possibility is that charge transfer reactions which were thermodynamically unfavorable at cell voltages above 1.1 V become favorable at lower cell voltage. The anode and cathode potentials were measured with respect to Ag/AgCl reference electrodes for the 10 mM and 20 mM BH_4^- cases. The results are plotted in Figure 4.19, where the potentials have been corrected to be *vs.* RHE. The potential of each Ag/AgCl reference electrode was measured *vs.* a normal hydrogen electrode (0.5 M H_2SO_4 and $P_{\text{H}_2} = 1$ ATM with a Pt mesh electrode) to obtain the correct offsets. Two horizontal dashed lines in Figure 4.19 demarcate the boundaries between H^+ stability and reduction in the oxidizer solution (a_1) and H_2O stability and reduction in the fuel solution (a_2). These lines are the same as line (a) in Figure 1.7, but in Figure 4.19 the potentials have been adjusted to reflect the concentrations in the experiment.

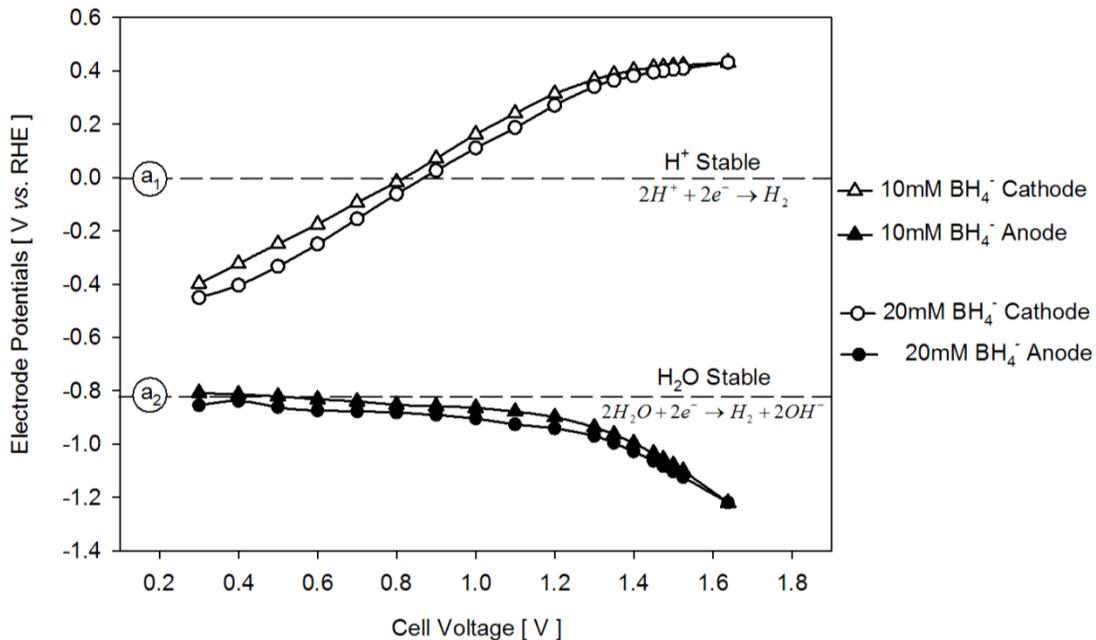


Figure 4.19. Anode and cathode potentials measured during baseline polarization curve, vs. Ag/AgCl reference electrodes and then corrected to RHE.

It is notable that proton reduction at the cathode is expected to begin at $V_{cell} = \sim 0.9$ V, which is roughly the point at which the slopes of the polarization curves change. Perhaps proton reduction (reaction R 1.23) becomes thermodynamically favorable when the cell voltage falls below ~ 1.1 V and augments the current density due to H_2O_2 reduction (reaction R 1.3).

In addition to providing a clue as to the processes dictating the current density, Figure 4.19 also provides insight into the relative magnitudes of the loss mechanisms at each electrode. At open circuit in the 20 mM BH_4^- case, the anode potential is -1.220 V vs. RHE and the cathode potential is $+0.431$ V vs. RHE. The anode potential is reasonable given the standard reduction potential for reaction R 1.2 (-1.24 V vs. RHE), but the cathode potential is 1.29 V less positive than the reduction potential for reaction R 1.3 corrected to local conditions (1.717 V vs. RHE). The

large disparity between the predicted and actual potential of the cathode at open circuit suggests other reactions are influencing the cathode overpotential. The loss of cathode potential is the primary reason the measured open circuit voltage (~ 1.61 V) is much lower than the open circuit voltage predicted by thermodynamics (3.01 V). Open circuit voltages in the range 1.5 to 1.7 V have been observed in all reported DBFC experiments using alkaline NaBH_4 and acidic H_2O_2 reactants.

As the cell voltage is decreased from open circuit, the anode potential in Figure 4.19 rises quickly while the cathode potential changes little. At higher current density, the anode potential changes little and the cathode potential becomes less positive in proportion to the decrease in cell voltage. The changes in electrode potential from open circuit to high current density show that the majority of activation overpotential originates at the anode and the majority of concentration overpotential occurs at the cathode. The greater anode activation overpotential agrees with the consensus in the literature that BH_4^- oxidation in alkaline media is slower than H_2O_2 reduction in acidic media. The difference in concentration overpotential is not surprising given the differences in the transport parameters for BH_4^- and H_2O_2 discussed previously.

Measurement of the 1 mM, 2.5 mM and 5 mM BH_4^- polarization curves was halted at 0.4 V because lower cell voltages produced declining current density, which suggested that the load would drive the cell if the voltage were further decreased. All curves plotted in Figure 4.17, and Figure 4.18 were truncated at 0.4 V for consistency, yet the 10 mM and 20 mM BH_4^- measurements proceeded to 0.3 V. The full 10 mM and 20 mM BH_4^- curves are plotted in Figure 4.20, where it is clear that

the greater BH_4^- concentration in the 20 mM curve led to a sudden increase in current density at low cell voltage. This feature was consistent among all three of the 20 mM BH_4^- measurements (see the error bars in Figure 4.20) and in the polarization curves of Figure 4.15.

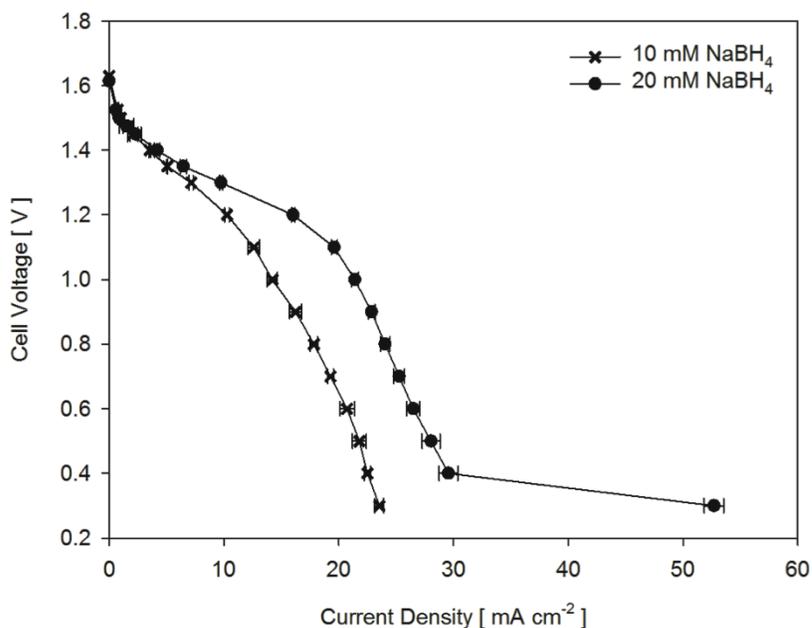


Figure 4.20. Plots of the entire 10 mM and 20 mM BH_4^- polarization curves.

These measurements support the hypothesis that additional reduction reaction takes place at the cathode. It may be reaction R 1.23, which should behave as observed when the cathode potential falls below 0.00 V vs. RHE. The current density at cell voltages from 1.1 to 0.6 V may be dictated by the H_2O_2 transport limit, and then at lower voltages, proton reduction begins to supply additional current density. Since there is ample BH_4^- at the anode which was underutilized at the H_2O_2 transport limit, and ample H^+ at the cathode, the current density rises rapidly once H^+ reduction

begins. The lack of a hard transport limit may be due to the current contributed by H^+ reduction increasing as the current contributed by H_2O_2 becomes transport limited.

A comparison between the 20 mM BH_4^- polarization curves in Figure 4.15 and Figure 4.20 can shed additional light on the transition from H_2O_2 reduction to H^+ reduction, by showing how membranes with different histories can influence the onset of H^+ reduction. A single membrane was used for all of the setup and model calibration experiments; it is labeled “Original Membrane” in Figure 4.21. A fresh membrane was used for the experiment examining the effects of voltage stepping direction; it is labeled “Fresh Membrane” in Figure 4.21.

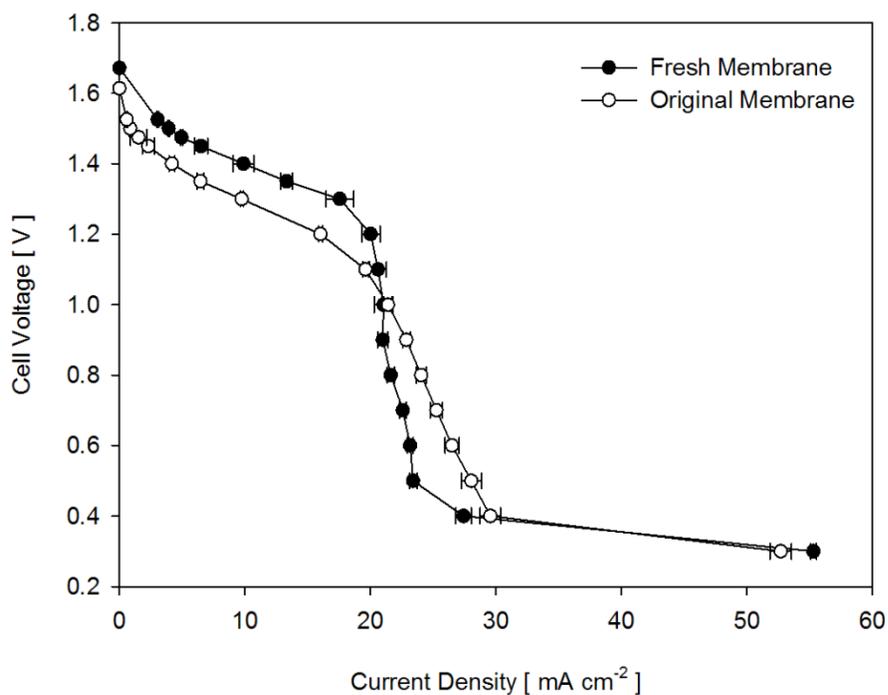


Figure 4.21. 20 mM BH_4^- polarization curves; “Original Membrane” is replotted here from the suite of five model calibration curves and “Fresh Membrane” is replotted here from the hysteresis experiment.

The fresh membrane showed a steeper drop in current density at $\sim 21 \text{ mA}\cdot\text{cm}^{-2}$, whereas the original membrane transitioned more gradually. Comparing cathode potential measurements from the two cases showed that the cathode was more negative with the original membrane, presumably because the original membrane had been degraded by the setup experiments and therefore incurred a greater ohmic drop. The greater membrane ohmic drop pulled the cathode to lower potential at each current density, causing the cathode to reach the onset potential for H^+ at lower current density. The onset of H^+ reduction at lower current density blurred the transition from H_2O_2 reduction to H^+ reduction, yielding the gradual transition for the original membrane. This explanation is bolstered by modeling results in Chapter 5 which predict a H_2O_2 transport limited current density of $\sim 21 \text{ mA}\cdot\text{cm}^{-2}$.

Polarization data were plotted in the ideal cell analysis of Chapter 3 as cell power density vs. cell voltage because it provided a clear picture of the fuel cell operating space. Power curves from the experiments are shown in Figure 4.22, which have shapes and trends similar to the ideal case power curves in Figure 3.12. The ideal and measured power curves differ, however, in that peak power shifts to lower cell voltage with increasing borohydride concentration in Figure 3.12 and not in Figure 4.22. The shift in the ideal case analysis was due to greater ohmic losses at higher current density, which depressed the cell voltage. The trend may be absent in the measurements because the measured average current densities were much lower than in the ideal case analysis, so the ohmic overpotentials were less evident.

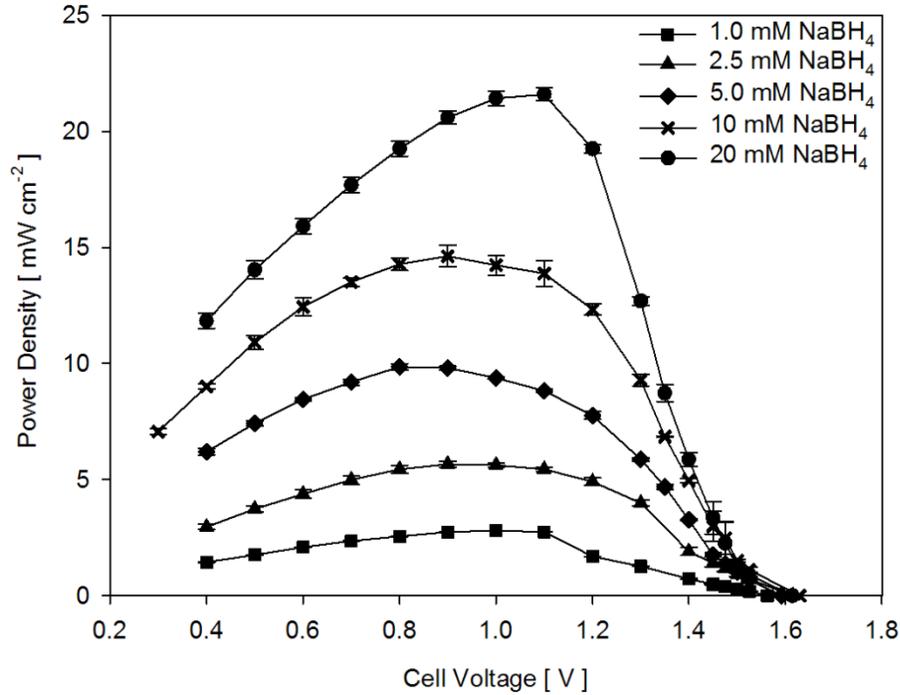


Figure 4.22. Measured power curves with varying BH_4^- concentration in 2 M NaOH. In all cases, the oxidizer was 40 mM H_2O_2 / 1 M H_2SO_4 . Fuel and oxidizer flow rates were both $10 \text{ mL} \cdot \text{min}^{-1}$.

The model calibration polarization curves in Figure 4.17 were measured with reactant concentrations lower than those often reported in DBFC experiments. The lower concentrations were chosen to minimize the rates of BH_4^- hydrolysis and H_2O_2 decomposition, which at high rates can produce large gas volumes. When the reactant flows contain large volume fractions of gas, the incompressible liquid assumption breaks down and a multi-phase flow model becomes necessary. A multi-phase flow model was beyond the scope of this study. Omitting multiphase flow in early models of new fuel cell chemistries is not unprecedented; early PEMFC models neglected liquid water transport and early DMFC models neglected CO_2 in the aqueous fuel for similar reasons. The complexity of multi-phase flow was added in

advanced PEMFC and DMFC models, and it may be feasible for future DBFC modeling.

Despite the complexities of modeling DBFC performance with higher reactant concentrations, one polarization curve was measured with moderate concentrations of 50 mM NaBH₄ / 2 M NaOH fuel and 250 mM H₂O₂ / 1 M H₂SO₄. Figure 4.23 shows the polarization and power curves measured for these operating conditions. This polarization curve exhibits the same gradual decline in cell potential with current density that appeared in the high borohydride concentration curves of Figure 4.17. One feature of the polarization curve in Figure 4.23 which stands out in comparison to the polarization curves of Figure 4.17 is an additional change in slope beginning at 0.5 V cell potential. This change in the curve may be the beginning of a BH₄⁻ transport limit.

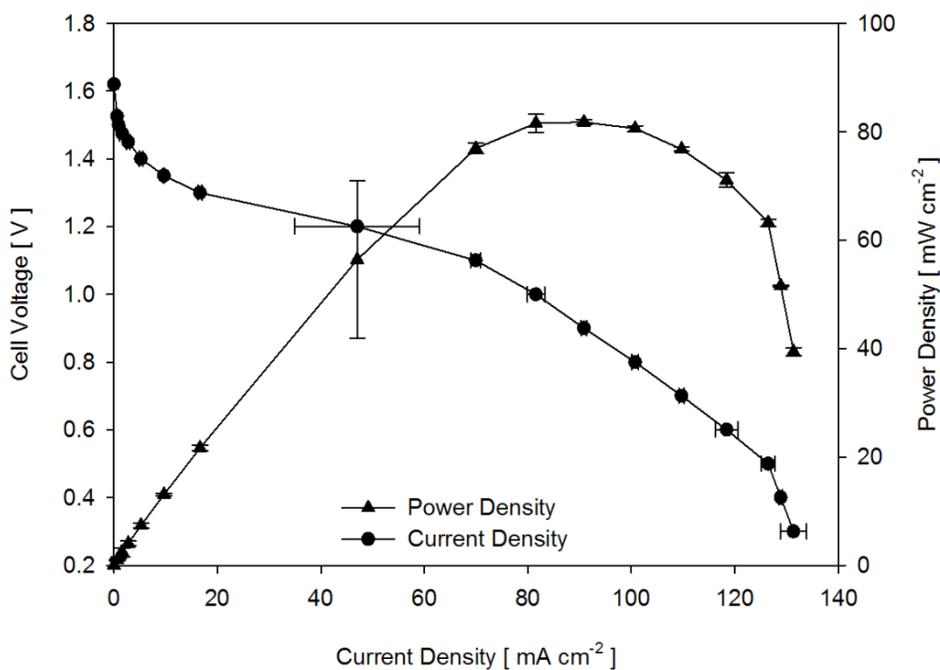


Figure 4.23. Polarization and power curves measured for 50 mM NaBH₄ / 2 M NaOH fuel and 250 mM H₂O₂ / 1 M H₂SO₄ oxidizer. Both flow rates were 10 mL·min⁻¹.

4.5.2 Gas Production Observations

The DBFC used to measure polarization curves did not permit direct observation of the electrodes during operation, but the electrode potential measurements did permit subsequent *ex situ* experiments which reproduced the electrode behavior where it could be observed. Such experiments were conducted to better understand the relationships between cell potential and gas production. In each *ex situ* experiment, one of the graphite plates from the DBFC was submerged in a beaker of solution having the same concentrations as in the cell experiments. A Ag/AgCl reference electrode and a Au counter electrode were also submerged in the beaker solution. The counter electrode had an area more than 10 times the area of the electrode on the graphite plate so that the observed behavior would be determined by the electrode on the graphite plate. The graphite plate electric potential was cycled through the range observed in the cell experiments by an AutoLab PGSTAT 30 potentiostat/galvanostat.

The cell cathode was submerged in 40 mM H₂O₂ / 1 M H₂SO₄ and cycled from 0.4 V to -0.5 V vs. RHE at 0.01 V·s⁻¹. The open circuit potential in the *ex situ* experiment was 0.426 V, which agrees with the measured cathode potential in the assembled cell at open circuit (see Figure 4.19). At open circuit, rapid gas production was observed at the electrode (but not elsewhere on the graphite plate). A subsequent experiment omitting H₂O₂ from the solution did not exhibit this behavior, confirming that H₂O₂ was involved in gas production, which was likely O₂ produced by H₂O₂ decomposition. H₂O₂ decomposition appeared to cease immediately when the graphite plate potential was decreased from open circuit. Visible gas production

(again, only at the electrode) resumed when the potential reached approximately -0.48 V vs. RHE. H₂ production via reaction R 1.23 was likely the source of gas at low potential, because reaction R 1.23 is thermodynamically favorable at potentials more negative than ~0.00 V vs. RHE in 1 M H₂SO₄. The current density also increased substantially when H₂ production began, although this observation cannot be used quantitatively because the entire plate (not just the electrode) was submerged in solution. These *ex situ* observations corroborate observations of gas bubbles in the oxidizer effluent line, in the polarization curve measurements at open circuit and 0.3 V cell voltage.

The same *ex situ* experiment was carried out with the DBFC anode in a beaker containing 20 mM NaBH₄ / 2 M NaOH. The anode was cycled from -1.3 V to -0.7 V vs. RHE at 0.01 V·s⁻¹. The open circuit potential in the *ex situ* experiment was -1.115 V vs. RHE, which is similar to the value observed during the polarization curve measurements at open circuit (-1.22 V vs. RHE). The difference may be due to rapid gas production observed at open circuit in the beaker experiment. The gas was presumably H₂ produced by BH₄⁻ hydrolysis via reaction R 1.4. Since the solution was quiescent in the beaker and flowed in the cell, the concentration of BH₄⁻ near the electrode may have fallen further in the *ex situ* experiment as BH₄⁻ was consumed by hydrolysis. Lower BH₄⁻ concentration would have shifted the equilibrium potential to a less negative value, as observed. The current density *and* gas production rate in the *ex situ* experiment were observed to increase as the cell potential was made less negative. The greater rate of H₂ production at high current density is not an expected outcome for the anode mechanism consisting of reactions R 1.2 and R 1.4, but it does

agree with mechanisms which assume that H^* plays a role in determining the anode reaction rates, such as that proposed by Rostamikia [36]. BH_4^- should adsorb more readily on a less negative anode, raising the surface concentration of BH_4^- ($\theta_{BH_4^-}$). Other studies have shown that the rate of BH_4^- adsorption and surface coverage of H should both increase as the anode becomes less negative [36, 70]. As $\theta_{BH_4^-}$ grows, the surface concentration of H^* (θ_H) will also grow due as BH_4^- dehydrogenates. Greater θ_H has two results; first, higher current density as OH^- from solution reacts with H^* to form H_2O and provide an electron to the anode (reaction R 1.15), and second, greater H_2 production via reaction the Heyrovsky reaction (R 1.27) and/or the Tafel reaction (R 1.28).

The model calibration experiments did not include quantitative measurements of gas production rate, yet several qualitative observations were possible. On the oxidizer side, initial setup experiments were conducted with high (1 M) H_2O_2 concentration before lower concentrations were chosen for the model calibration experiments. High H_2O_2 concentration yielded prodigious gas at open circuit which was evident by gas volume fractions of $\sim 1/3$ in the oxidizer effluent stream. The lower (40 mM) H_2O_2 concentration used for the model calibration experiments produced fewer bubbles. On the fuel side, bubbles were evident for all operating conditions and inlet BH_4^- concentrations. The gas volume fraction in the fuel effluent was clearly larger for higher BH_4^- concentration, lower NaOH concentration, and at high current density.

4.5.3 Other Measurements and Observations

A Nafion membrane was chosen in part because it resists attack by both acids and bases, which is an important property for DBFCs which can have pH gradients of ~ 14 across the membrane. Some researchers have suggested that the pH gradient may drive H^+ and/or OH^- through the membrane if the acid and base concentrations are sufficiently high [7, 92], despite the electric potential gradient across the membrane opposing entry of either species and the Donnan potential opposing anion entry. The pHs of the fuel and oxidizer solutions were measured before and after flowing through the cell to ascertain whether or not significant crossover was taking place. An Orion model 720A pH meter was used to measure the pH of fuel and oxidizer before and after flowing through the cell in a baseline polarization curve experiment. The effluent was collected while the cell was at open circuit, which is when the orientation of the electric field in the membrane was expected to yield the largest crossover rates. The pH meter was calibrated to a pH 10.0 buffer prior to the fuel measurement and a pH 4.0 buffer prior to the oxidizer measurement. The results were pH = 13.795 for fuel before and after flowing through the cell, indicating little or no crossover of H^+ from the oxidizer solution. The oxidizer measurements were pH = 0.295 before and 0.302 after – an insignificant change given the pH meter accuracy. Thus no crossover induced changes in pH was detected.

The DBFC model described in Chapter 2 includes the estimation of solution mass densities based on the apparent molar volumes of the solutes. The densities of the 50 mM BH_4^- / 2 M NaOH fuel and 250 mM H_2O_2 / 1 M H_2SO_4 oxidizer solutions were measured to provide a basis on which to judge the model accuracy in this

regard. These concentrations were chosen for the density measurements because any discrepancy between the measurement and the model should be largest in this case; lower concentrations will yield solution densities closer to that of pure water.

The solution densities were measured by dispensing 5 mL onto a weigh boat with a Finnpiette micropipette. The solution mass was measured by a Denver Instruments M-220D microbalance, and then the dispensed volume and mass were used to calculate a mass density. This procedure was repeated 10 times each for 18 MΩ water, fuel solution and oxidizer solution, all equilibrated to a 23°C laboratory. The results are provided in Table 4-2. The measured value for the density of water was 1.2% lower than the value provided by NIST [93]. The discrepancy may be due to systematic error (for example, less than 5 mL dispensed) or dissolved gasses in the water which the NIST value omits. The densities predicted by the model for the fuel and oxidizer solutions both exceed the measured values by 2.5%. In these cases as well, some or all of the discrepancy could be due to the model neglecting dissolved gases, predominantly N₂.

Table 4-2. Density measurements and comparison to model predictions for fluids at 23°C. Fuel: 50 mM NaBH₄ / 2 M NaOH, Oxidizer: 250 mM H₂O₂ / 1 M H₂SO₄.

Fluid	Measured	Predicted	NIST	Discrepancy
Water	0.9852	-	0.9975	-1.2%
Fuel	1.0675	1.0937	-	+2.5%
Oxidizer	1.0486	1.0747	-	+2.5%

4.5.4 Conclusions

Each loss mechanism in the cell corresponds to a different aspect of DBFC operation which the model must capture to accurately predict cell performance. The

activation overpotential originates with the electrode reactions, the ohmic overpotential is predominantly a result of transport in the membrane and the concentration overpotentials arise from transport in the channels. The polarization curves in Figure 4.17 should be a good test of the model because they exhibit all three loss mechanisms. Furthermore, experiments varying the cell stoichiometry should test these processes on both the fuel and oxidizer sides of the membrane.

Figure 4.17 indicates that the transport limiting species was BH_4^- when the BH_4^- concentration was 1 mM and 2.5 mM. Figure 4.18 and Figure 4.19 suggest that at higher concentrations of BH_4^- the transport limit shifted to the cathode, and the process became more complex. The current density in these cases approaches the transport limit gradually, indicating that migration, a potential dependent charge transfer reaction, or the onset of another charge transfer reaction is involved.

Figure 4.19 also shows that most of the activation overpotential originates at the anode, and that the depressed open circuit voltage (compared to the theoretical cell voltage) is due to a shift in the cathode potential. Qualitative observations indicate that the rate of cathode O_2 production in the model calibration experiments was minor, and that H_2 production at the anode increased with increasing current density. The trend relating H_2 production to current density supports anode reaction mechanisms in which the anode reactions are related by a shared pool of surface adsorbed hydrogen.

pH measurements support the model assumption that OH^- and H^+ do not cross the membrane, at least for the conditions examined. The measurements of solution mass density indicate that the model predicts the densities of the fuel and oxidizer

solutions to < 2.5% error, and some fraction of the error may be attributable to dissolved gases which are not included in the model.

Single-cell experiments yielded valuable insight into the factors dictating DBFC performance. They also provided the measurements necessary for model calibration, yet the measurements alone would have been insufficient. The experiments also guided reaction mechanism selection by showing the importance of BH_4^- hydrolysis at the anode and H_2O_2 decomposition at the cathode. These reactions strongly influence OCV, transport limit onset, and net current density by competing with the charge transfer reactions. Hydrolysis and decomposition must be included in a realistic DBFC analysis. Furthermore, H^+ reduction at the cathode was considered when calibrating the model because the experiments indicated it was likely to occur and shift the peak power density of the cell to higher current density.

Chapter 5: Model-Based Analysis of a Realistic DBFC

5.1 *Goals and Approach*

The ideal DBFC analysis in Chapter 3 revealed some of the trends linking cell design and operating conditions to performance, yet the ideal analysis was limited to transport-related phenomena by the lack of realistic reaction rates or competing electrode reactions. The transport-related trends provide useful design guidance, but the model must capture electrode reactions in a more realistic way if we are to obtain a complete picture of DBFC performance.

The modeled electrode reactions were made more realistic by calibrating the rate parameters to measurements from Chapter 4. The calibration process involved two steps: choosing an appropriate reaction mechanism for each electrode based on analysis of insight from the literature and the experimental tests, and then fitting the uncertain rate parameters to our measurements. The calibrated model was then used to examine the sources of efficiency loss in a realistic DBFC and the ways in which these losses depend on cell design and operating conditions. The goal of this analysis was to improve DBFC design by recommending loss mitigation strategies and guiding future research efforts. The DBFC model provided insight which would have been difficult to obtain through experiments alone.

5.2 *Reaction Mechanism Fits to the Measurements*

Because the measurements in Chapter 4 only provided global cell performance, global rate expressions were used to capture the essential features of the complex

electrode reactions. In this effort to fit the data, one anode mechanism and two cathode mechanisms were fitted to the measurements. The anode mechanism consisted of R 1.2 (BH_4^- oxidation) and R 1.4 (BH_4^- hydrolysis). The first cathode mechanism consisted of R 1.3 (H_2O_2 reduction) and R 1.5 (H_2O_2 decomposition). These mechanisms were selected because they are promulgated widely in the literature as capturing the essential features of DBFC electrode reactions. The fit quality was improved by a second cathode mechanism, which added R 1.23 (H^+ reduction). R 1.23 was selected because the experiments of Chapter 4 indicated it occurs at low cell potential, where the fit from the first mechanism was most in error.

The DBFC used for experiments in Chapter 4 was designed specifically for model calibration, and hence, adapting the model to accurately reflect the real cell was straightforward. Model parameters used for the fitting process included the cell geometry and operating conditions listed in Table 5-1.

Table 5-1. Model parameters for fitting process, taken from the experiments in Chapter 4. Both channels shared the same dimensions.

Geometric		Operating	
Channel Length	50 mm	Fuel Flow Rate	$10 \text{ mL} \cdot \text{min}^{-1}$
Channel Depth	0.50 mm	Oxidizer Flow Rate	$10 \text{ mL} \cdot \text{min}^{-1}$
Channel Width	5.0 mm	Temperature	23°C
Membrane Thickness	$208 \mu\text{m}$ [94]	Oxidizer Concentrations	$40 \text{ mM H}_2\text{O}_2 /$ $1 \text{ M H}_2\text{SO}_4$

Several simplifications employed in the ideal DBFC analysis were valid for the model calibration. For example, the model assumed that momentum boundary layers were fully developed at the inlets, which was reasonable given that the DBFC channels in the experiments extended beyond the catalyst layers with ample length to ensure fully developed flow. As another example, the model assumed channel walls

were far apart so that relevant state variables varied in the x - and y -directions only; this was reasonable given the 10:1 aspect ratio of the DBFC channels.

5.2.1 Fitting Approach and Procedure

Rate parameters and electrode roughnesses were fitted to the measurements for each reaction mechanism with the goal of obtaining the “best” fit to the five measured polarization curves. Best can be defined in many ways; in this case it refers to minimizing the 2-norm of errors between the measured and predicted current density at each cell voltage on the polarization curves.

An error function was developed which repeatedly called the main DBFC model code to calculate the differences between measured and predicted current densities. The error function output a vector \vec{e} in which each element was the error between one measurement and the corresponding model prediction. In some cases the model predictions were compared to a subset of the measurements, and in other cases the model predictions were compared to the entire data set (all five measured polarization curves). The error function was called by the MATLAB function *lsqnonlin*, which used a Newton search approach to minimize $\|\vec{e}\|$. The “trust region reflective” algorithm was chosen because it respects bound constraints on the fitted parameters. Fitted reaction rate constants k were constrained to the interval $(0,\infty)$, symmetry factors β were constrained to $(0,1]$ and roughness factors ℓ were constrained to $[1,30]$. This approach does not guarantee a global minimum $\|\vec{e}\|$; it is possible to find a local minimum. Trial and error showed that the initial guess must produce a polarization curve differing from the measurements by less than one order of magnitude, and share the same trend as the measurements, or the fitting algorithm may find a local

minimum. At least three widely spaced starting guesses were evaluated in each fitting effort, and the fit was not accepted as “final” until all three guesses resulted in the same fit, suggesting the fit *may* be global.

5.2.2 Fitting the Simplest Reaction Mechanism

The rates of charge transfer reactions were estimated by Eq. 1.11 with fitting parameters k_a and β_a . Parameters n_e , β_c and k_c in Eq. 1.11 were not fitted. The number of electrons transferred in the rate determining step for each reaction was assumed to be $n_e = 1$ and the cathodic direction symmetry factors were assumed to be $\beta_c = (1 - \beta_a)$. These are both common assumptions as discussed in Chapter 1. The values of k_c were chosen to ensure thermodynamically consistent rate equations, i.e. they would predict zero net rate under standard conditions when the electrode-interface electric potential difference $\Delta\phi$ was equal to the equilibrium value E_{rxn}^0 . These values were found by setting $a_k^{vk} = 1$, $T = 298$ K, $\Delta\phi = E_{rxn}^0$ and $r = 0$ in Eq. 1.11 and solving for k_c . This process was repeated for each set of parameters k_a and β_a evaluated by the fitting algorithm, so that thermodynamic consistency was maintained despite changes to k_a and β_a .

This approach to establishing thermodynamic consistency can also be cast in terms of the reaction equilibrium constant K_{eq} . The relationships between k_a , k_c , K_{eq} and $\Delta\mu_{rxn}^0$ were discussed in Chapter 1, where Eq. 1.13 (shown here for convenience) related the equilibrium constant to an Arrhenius-type activation energy barrier. At equilibrium under standard conditions, $\Delta\mu_{rxn} = FE_{rxn}^0$, so Eq. 1.13 gives $k_c = k_a e^{fE_{rxn}^0}$.

$$k_a/k_c = K_{eq} = e^{-\Delta\mu_{rxn}/RT} \quad \text{Eq. 1-13}$$

Charge transfer reactions were written in terms of the fitting parameters k_a and β_a by substituting for k_c and β_c in Eq. 1.11. The anode and cathode rate equations (Eq. 5.1 and Eq. 5.2) omit the concentrations of OH^- , H^+ and H_2O because they are present in excess, and therefore have little influence over the rates.

$$r_{R1.2} = k_{a,R1.2} C_{\text{BH}_4^-} e^{\beta_{a,R1.2} f \Delta\phi} - k_a e^{f E_{R1.2}^0} C_{\text{BO}_2^-} e^{-(1-\beta_{a,R1.2}) f \Delta\phi} \quad \text{Eq. 5.1}$$

$$r_{R1.3} = k_{a,R1.3} e^{\beta_{a,R1.3} f \Delta\phi} - k_a e^{f E_{R1.3}^0} C_{\text{H}_2\text{O}_2} e^{-(1-\beta_{a,R1.3}) f \Delta\phi} \quad \text{Eq. 5.2}$$

The rates for chemical reactions occurring at each electrode were estimated by first order rate expressions assuming irreversibility:

$$r_{R1.4} = k_{f,R1.4} C_{\text{BH}_4^-} \quad \text{Eq. 5.3}$$

$$r_{R1.5} = k_{f,R1.5} C_{\text{H}_2\text{O}_2} \quad \text{Eq. 5.4}$$

The rate parameters were initially fitted only to the activation regions of the polarization curves because the reaction rates influence these regions most. This approach was expected to strongly couple the fitting errors to the reaction rate parameters and drive the fitting algorithm toward a solution quickly. For the purposes of fitting, activation regions were said to encompass cell potentials from open circuit to 1.1 V. The values listed in Table 5-2 gave the best fit. The measured and predicted polarization curves are shown in Figure 5.1.

Table 5-2. Fitted reaction rate parameters assuming R 1.2 and R 1.4 at the anode and R 1.3 and R 1.5 at the cathode.

Anode Parameters		Cathode Parameters	
$k_{a,R1.2}$	$9.25 \times 10^{-3} \text{ m} \cdot \text{s}^{-1}$	$k_{c,R1.3}$	$7.54 \times 10^{-3} \text{ m} \cdot \text{s}^{-1}$
$\beta_{a,R1.2}$	0.098	$\beta_{c,R1.3}$	0.455
$k_{f,R1.4}$	$3.09 \times 10^{-4} \text{ m} \cdot \text{s}^{-1}$	$k_{f,R1.5}$	$6.34 \times 10^{-4} \text{ m} \cdot \text{s}^{-1}$
ℓ_a	2.73	ℓ_c	4.11

The fitted rate parameters are not directly comparable to the values reported by Santos [60] and Finkelstein [39, 62], because these authors reported rates in terms of overpotential rather than the electrode-interface potential differences used here. Furthermore, Finkelstien calculated overpotentials by assuming the equilibrium potential for each reaction was equal to the observed onset potential. Nevertheless, the fitted rate parameters are similar to the reported values. Finkelstein reported $k_{a,R1.2} = 6.2 \times 10^{-4} \text{ m} \cdot \text{s}^{-1}$ and $\beta_{a,R1.2} = 0.2$ on Au for 5 mM NaBH₄ in 1 M NaOH, and $k_{c,R1.3} = 8 \times 10^{-3} \text{ m} \cdot \text{s}^{-1}$ and $\beta_{c,R1.3} = 0.45$ on Pt for 5 mM H₂O₂ in 0.5 M H₂SO₄.

Figure 5.1 shows good agreement between the model and the 1, 2.5 and 5 mM BH₄⁻ curves, with R^2 values of 0.920, 0.986 and 0.996 respectively. The activation and ohmic overpotential dominated regions of all five curves are well described by the model, but discrepancies appear at high current density in the 10 and 20 mM BH₄⁻ curves. Two discrepancies are apparent. First, the predicted current density for the 10 mM BH₄⁻ curve is too high in the cell voltage range 0.5 to 1.1 V. Second, the hard transport limit at $\sim 21 \text{ mA} \cdot \text{cm}^{-2}$ does not match the gradual decline in current density shown by the 10 mM and 20 mM BH₄⁻ curves.

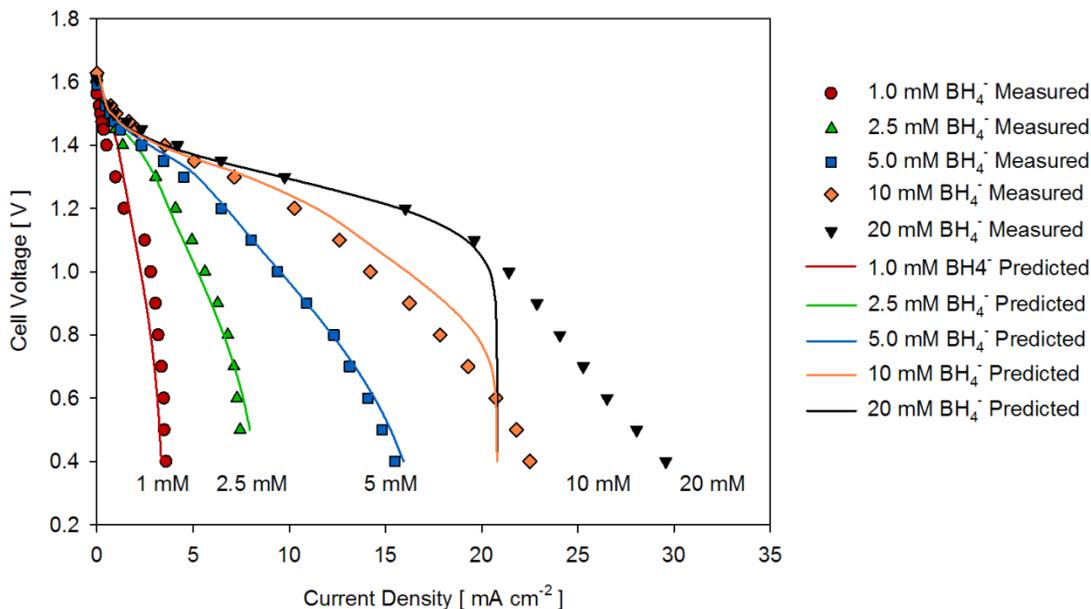


Figure 5.1. Comparison between measured and predicted polarization curves, with the model fitted to reaction mechanisms consisting of R 1.2 through R 1.4.

The predicted BH_4^- and H_2O_2 concentrations in the 20 mM BH_4^- case are plotted in Figure 5.2 for a cell voltage of 0.4 V, which shows that the predicted concentration of H_2O_2 approaches zero at the cathode interface. Thus, the predicted $\sim 21 \text{ mA}\cdot\text{cm}^{-2}$ limiting current density is imposed by H_2O_2 transport. This result agrees with measurements from Chapter 4 which showed a sudden decrease in cell voltage at $\sim 21 \text{ mA}\cdot\text{cm}^{-2}$ when the stoichiometry was H_2O_2 limited. Agreement on the limiting current density suggests the model accurately predicts the rate of H_2O_2 transport to the cathode.

The fitting process was repeated with the entire measured data set, but with little improvement in the fit. Weighting errors more heavily in the cell potential range between 0.6 and 1.0 V to emphasize agreement at the onset of the transport limit also did not improve the fit.

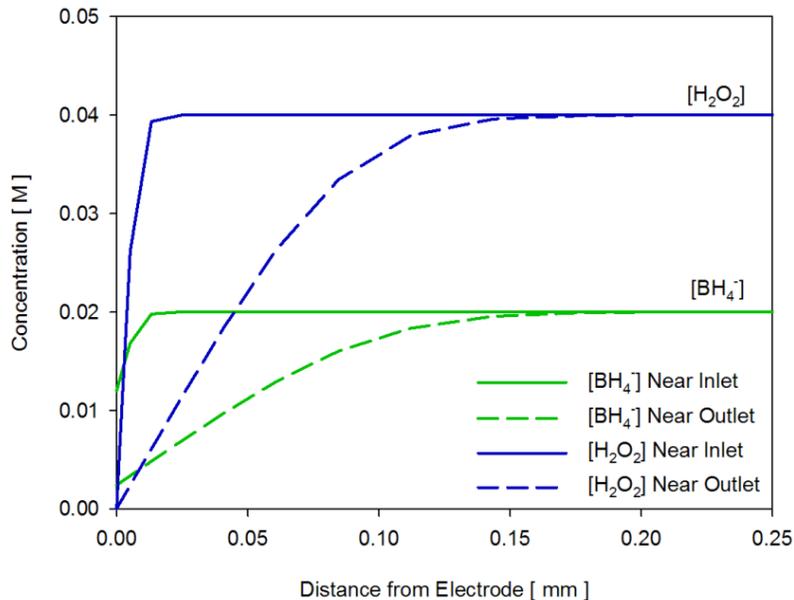


Figure 5.2. Predicted development of BH_4^- and H_2O_2 concentration boundary layers at the anode and cathode in the 20 mM BH_4^- case. Solid lines are near the inlet and dashed lines are near the outlet.

The electrode potentials may expose the source(s) of disagreement between the model and measurements. Figure 5.3 shows the measured electrode potentials and the predicted interfacial potential differences $\Delta\phi$ at each electrode. Surprisingly, the shapes of the measured and predicted curves are most similar at low cell potential, where the polarization curves differ most. The model predicts that the majority of cell voltage loss at open circuit occurs at the cathode and the concentration overpotential occurs almost entirely at the cathode, both in accordance with the measurements. The predicted relative rates of reaction between the anode and cathode at low current density differ from the measurements, as evidenced by Figure 5.3, where the predicted slopes of the electrode potential with respect to cell voltage near OCV are incorrect. Near OCV the anode potential should change rapidly (high activation overpotential) and the cathode potential should change slowly (low

activation overpotential) with respect to the cell voltage. These differences may introduce error when predicting the rates of competing reactions at low current density.

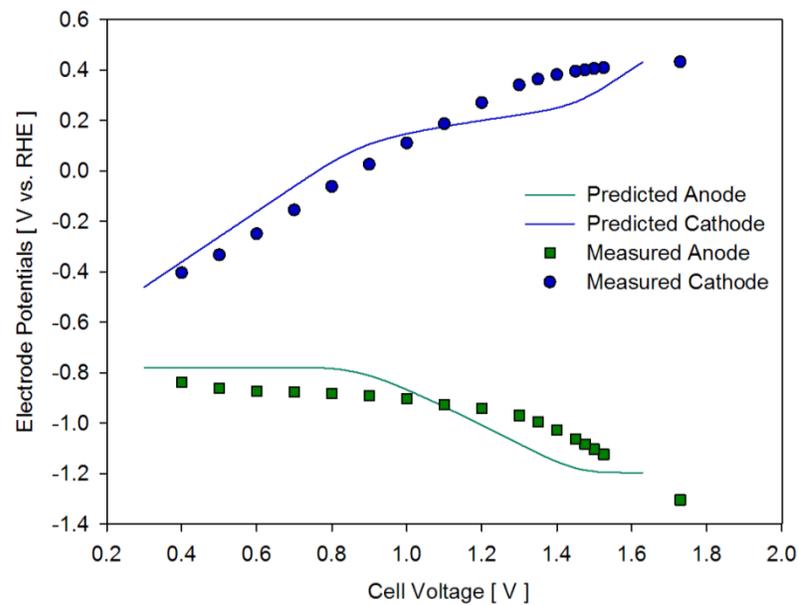


Figure 5.3. Measured electrode potentials vs. RHE and predicted interfacial electrode potential differences $\Delta\phi$ for the 10 mM BH_4^- baseline case.

The sources of error near open circuit may be due to phenomena which are omitted by the global reaction mechanisms, such as adsorption and surface reactions. For example, BH_4^- adsorption is understood to be slow and influence the rate of R 1.2 [36, 70]. The influence of adsorption would be strongest near open circuit where the anode is most negative, because the negative anode would discourage adsorption of BH_4^- anions. For the purposes of realistic DBFC analysis, the error near OCV was judged to be minor and these details were not added to the anode mechanism. The disagreement at low cell voltage could not be overlooked, however, because it

influenced the location of the peak power point. This discrepancy was addressed by adding H⁺ reduction to the cathode reaction mechanism.

5.2.3 Fitting a Mechanism Including Cathode H⁺ Reduction

As discussed in Chapter 4, the measured electrode potentials indicate that H⁺ reduction (R 1.23) is thermodynamically favorable in the 10 mM BH₄⁻ case for cell voltages below ~0.85 V and in the 20 mM BH₄⁻ case for cell voltages below ~0.80 V. These cell potentials correspond to onset of the erroneous hard transport limit in the model predictions. The correspondence between H⁺ reduction onset and the predicted transport limit suggested that H⁺ reduction may mitigate the disagreement between the measured and predicted 10 mM and 20 mM BH₄⁻ curves at low cell voltage.

The reaction mechanisms and rate expressions remained the same as in the first fitting effort, except for the addition of R 1.23 at the cathode and the corresponding rate in Eq. 5.5. Thermodynamic consistency was established for Eq. 5.5 using the same approach as in the first fitting effort. The second-order dependence on C_{H⁺} and n_e = 2 in Eq. 5.5 imply that the rate limiting step for R 1.23 is H⁺ approaching the cathode and accepting an electron, which must occur twice for the reaction to proceed. The concentration of H⁺ was included in the rate, despite H⁺ being in excess, because it was the only reactant in the rate equation and omitting it would have permitted a “runaway” reaction with no H⁺ present. While this is unlikely in the real cell, it could have caused numerical problems when solving the model.

$$r_{R5.1} = k_{c,R5.1} e^{-fE_{R5.1}^0} C_{H^+}^2 e^{2\beta_{a,R5.1} f \Delta \phi} - k_{c,R5.1} C_{H_2} e^{-2(1-\beta_{a,R5.1}) f \Delta \phi} \quad \text{Eq. 5.5}$$

Including R 1.23 at the cathode increased the current density at low cell potentials, although the fit still deviated from the measurements as shown in Figure 5.4. The complete set of fitted rate parameters is given in Table 5-3.

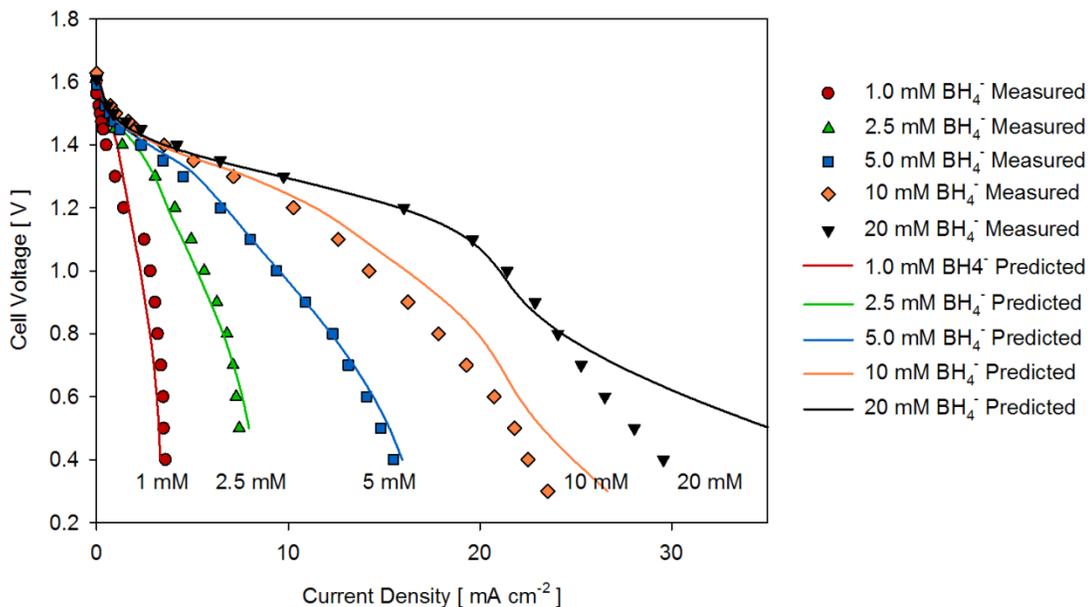


Figure 5.4. Comparison between measured polarization curves and those predicted by the model, assuming R 1.2 through R 1.4 and R 1.23 occur.

The large current density at low cell voltage in the 20 mM BH_4^- curve reflects the lower activation energy barrier to H^+ reduction as the cathode becomes less positive. A better fit could not be found with the reaction rate for H^+ reduction written as in Eq. 5.1. One way in which the reaction rate equation for H^+ reduction may be lacking is the absence of competition for catalyst surface sites. The mechanism of H_2O_2 reduction was discussed in Chapter 1, in which adsorbed OH plays a major role. The onset of significant H^+ reduction could be delayed to lower cathode potential by OH^* occupying sites on the Pd:Ir surface, although this study yielded no direct evidence to support this theory.

Table 5-3. Fitted reaction rate parameters assuming R 1.2 and R 1.4 at the anode and R 1.3, R 1.5 and R 1.23 at the cathode.

Anode Parameters		Cathode Parameters	
$k_{a,R1.2}$	$9.25 \times 10^{-3} \text{ m} \cdot \text{s}^{-1}$	$k_{c,R1.3}$	$7.54 \times 10^{-3} \text{ m} \cdot \text{s}^{-1}$
$\beta_{a,R1.2}$	0.098	$\beta_{c,R1.3}$	0.455
$k_{f,R1.4}$	$3.09 \times 10^{-4} \text{ m} \cdot \text{s}^{-1}$	$k_{f,R1.5}$	$6.34 \times 10^{-4} \text{ m} \cdot \text{s}^{-1}$
ℓ_a	2.73	$k_{c,R5.1}$	$1.19 \times 10^{-9} \text{ m}^4 \cdot \text{kmol}^{-1} \cdot \text{s}^{-1}$
		$\beta_{c,R5.1}$	0.141
		ℓ_c	4.11

No fuel cell operates over the entire the polarization curve; as discussed in Chapters 1 and 3, it is advantageous to operate at current densities up to (but not beyond) peak power. A model can diverge from the measurements beyond peak power point and still be useful, so long as it accurately captures the relationships between current density and cell potential up to the peak power point. The measured and predicted power densities are shown in Figure 5.5, where all five curves agree with the measurements to within 15% between OCV and 1.0 V. This range of cell potentials encompasses nearly all of the desirable operating range of this DBFC.

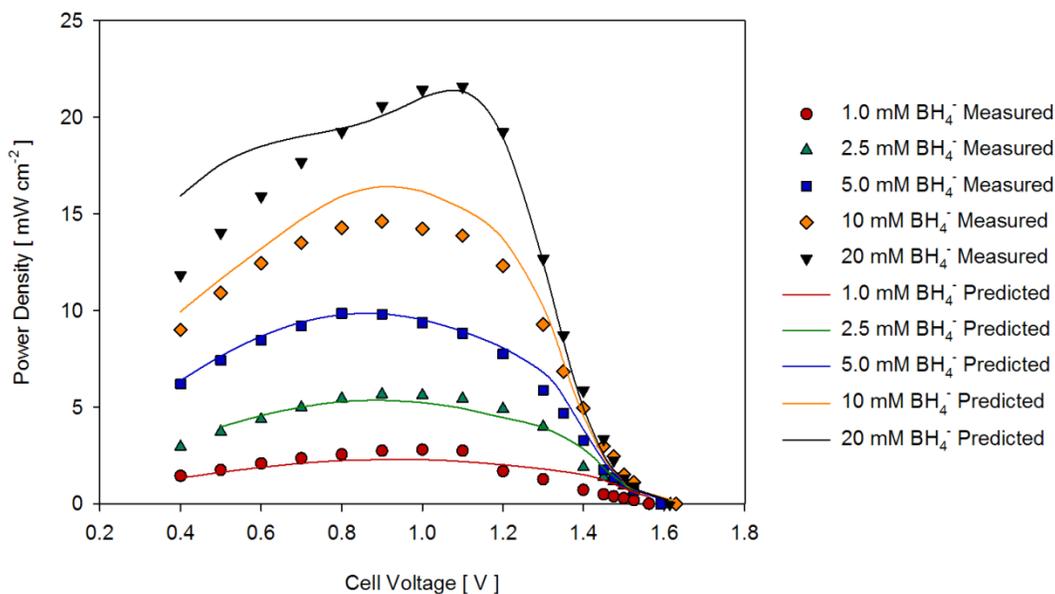


Figure 5.5. Comparison between measured power curves and those predicted by the model, assuming R 1.2 through R 1.4 and R 1.23 occur.

5.3 *Insights into Realistic DBFC Performance Provided by the Model*

The model fit with H⁺ reduction is not perfect, but it is sufficient for an analysis of realistic DBFC performance, particularly with the goal of identifying trends. Most DBFC experiments reported in the literature have used higher concentrations than those in the model calibration experiments of Chapter 4, so the realistic DBFC analysis was carried out with respect to a 50 mM BH₄⁻ / 250 mM H₂O₂ baseline to make it more similar to results in the literature. These concentrations were chosen because they constitute the “high concentration” case from Chapter 4, thus the model predictions can be compared to measurements, as shown in Figure 5.6. For the “high concentration” operating conditions, the model predicted polarization curve correlates to the measured curve with $R^2 = 0.980$. These concentrations yield a stoichiometry of approximately 1:1 in terms of transport rates to the electrodes, so the analysis should

reveal losses on both sides of the cell. Furthermore, this stoichiometry avoids H^+ reduction (which occurs in strongly oxidizer limited scenarios), keeping the results in a regime where the model accurately predicts cell performance.

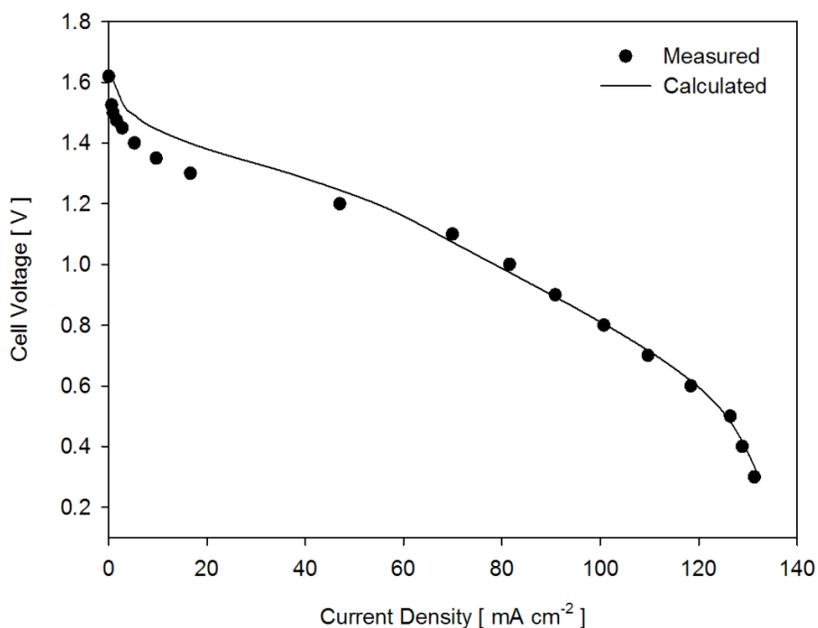


Figure 5.6. Comparison between measured and predicted polarization curves for the 50 mM BH_4^- / 250 mM H_2O_2 case with 10 mL min^{-1} flow rates.

5.3.1 Realistic DBFC Performance

The major difference between ideal and realistic DBFC operation is loss of reactants to BH_4^- hydrolysis and H_2O_2 decomposition. The predicted concentrations of H_2 and O_2 produced by these reactions are plotted in Figure 5.7 for the baseline case at 1.1 V cell. The rates of gas production and current density are plotted in Figure 5.8 with respect to position in the channel. The higher rate of H_2 production at the anode and greater diffusivity of H_2 in water leads to a larger concentration boundary layer as shown by Figure 5.7 (note the different concentration scales in this

figure). The gas production rates and current density show the same trends of diminishing magnitude with distance from the inlet, with each having substantially greater values near the inlet. These trends are the result of higher reactant concentrations at the inlet end of the electrode-solution interface, which drive charge transfer and chemical loss reactions more quickly there.

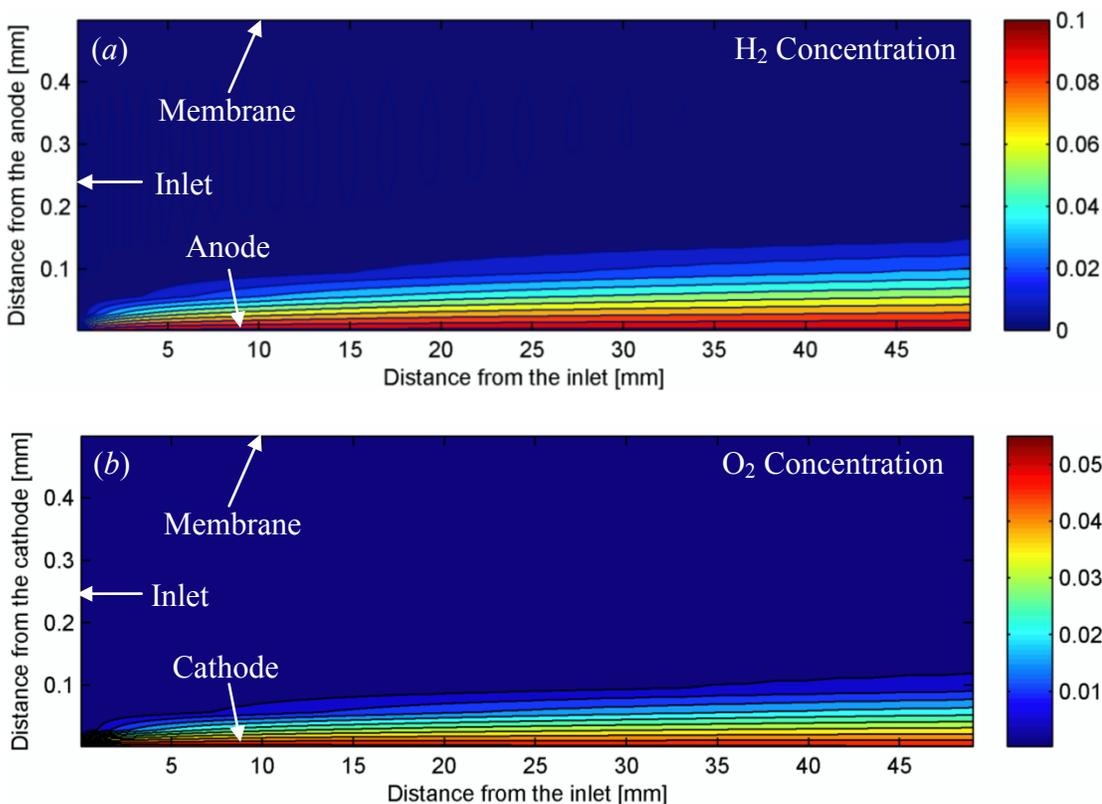


Figure 5.7. Predicted H₂ (a) and O₂ (b) concentrations in the channels for the baseline case at 1.1 V cell. Concentration units are mol·L⁻¹.

The predicted gas production rates are realistic because they depend on the fitted reaction rate parameters and transport of aqueous species, for which the diffusion, migration and advection fluxes can be estimated accurately. The predicted rates of H₂ and O₂ transport in the channel, however, are likely in error because the predicted

concentrations exceed the saturation limits for these species in water. The saturation limit of H_2 in pure water at 298 K is 7.8×10^{-4} M, and the saturation limit of O_2 in 1 M H_2SO_4 is 9.3×10^{-4} M [91]. Concentrations above these limits imply that bubbles will form, as observed in the experiments. The model does not include multiphase flows of liquid and gas bubbles (for reasons discussed in Chapter 2), and so the model predictions of gas concentration are useful only as an indicator of likely bubble generation locations. Most bubble formation should take place near the inlets due to the high concentrations and preponderance of gas production there; this insight could be useful for the design of graded catalyst structures which discourage bubble adhesion near the inlets and emphasize reactant access to the catalyst at points further from the inlet.

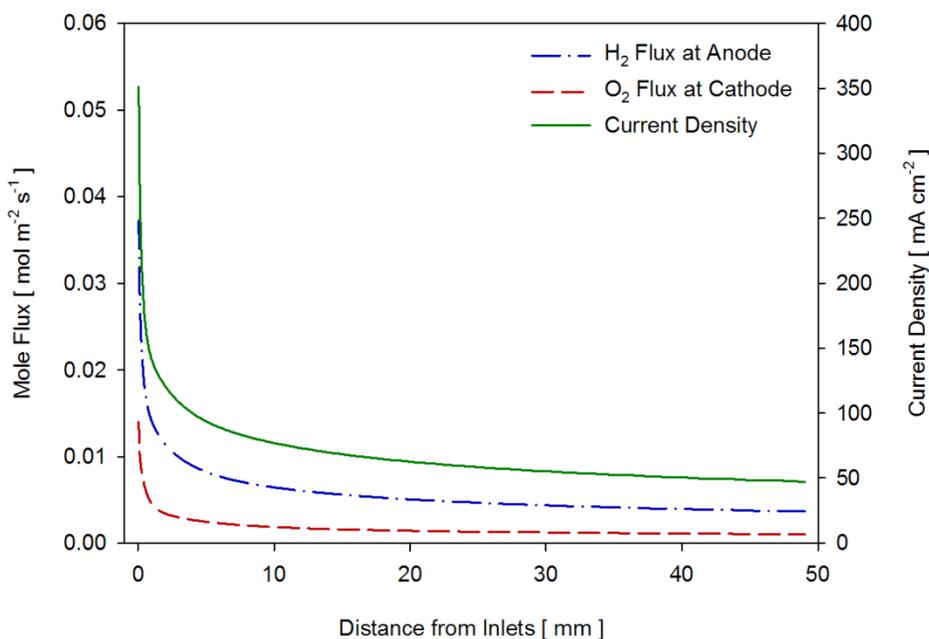


Figure 5.8. Predicted current density and H_2 and O_2 production rate distributions with respect to position in the channel, for the baseline case at 1.1 V cell.

The rate of gas production is only one metric for judging the impact of competing electrode reactions. Coulombic efficiency η_{ce} is the fraction of reactant reaching the electrode that participates in charge transfer reactions. Higher coulombic efficiency implies that fewer electrons are being lost to competing reactions. For example, complete electro-reduction of every BH_4^- anion reaching the anode would give an anode coulombic efficiency of 100%. Local coulombic efficiency at each electrode was calculated as the ratio of predicted current density i to the current density possible if all of the electrochemically active species k arriving at the electrode were consumed in charge transfer reactions (Eq. 5.6). Predicted coulombic efficiencies at each electrode are plotted in Figure 5.9 with respect to channel (x -) position.

$$\eta_{ce}(x) = \frac{i(x)}{\sum_k \vec{J}_k(x) v_e / v_k} \quad \text{Eq. 5.6}$$

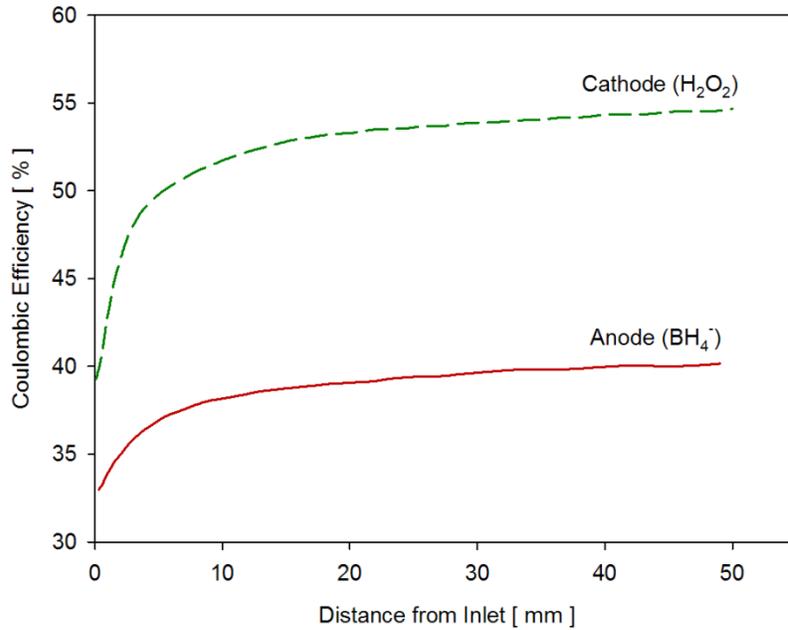


Figure 5.9. Predicted coulombic efficiencies at the anode and cathode with respect to distance from the channel inlets, for the baseline case at 1.1 V cell.

Figure 5.9 shows that for these operating conditions, the cathode delivers electrons to arriving H_2O_2 more efficiently than the anode extracts electrons from arriving BH_4^- . The cathode reduces electrochemically more than half of the arriving H_2O_2 and decomposes the rest to O_2 and H_2O . The anode electro-oxidizes less than 40% of the arriving BH_4^- . The coulombic efficiencies of both electrodes increase with distance from the inlets; since concentrations at the electrode interfaces fall and overpotentials grow with distance from the inlets. Figure 5.9 shows that these changes must favor charge transfer reactions. The low overall coulombic efficiencies and variation with channel position suggest that a priority for future research should be to shift the relative rates of reaction in favor of charge transfer reactions. One way to accomplish this shift is through novel catalyst materials and morphologies [42], but another is to operate the cell in a regime favoring charge transfer reactions.

Cell voltage strongly influences DBFC behavior and therefore presents an opportunity to tailor cell operation to more efficiently convert reactant chemical energy into electricity. In the parlance of the controls community, cell voltage is a powerful lever with which to adjust cell operation. The coulombic efficiencies of the anode and cathode are plotted with respect to cell potential in Figure 5.10, which shows that lower cell potentials raise the coulombic efficiencies of both electrodes. In effect, lower cell potential decreases the activation energy barriers to charge transfer reactions at both electrodes as the anode becomes less negative and the cathode less positive. The rates of charge transfer reactions increase, but because the rates of chemical hydrolysis and decomposition in the mechanism of section §5.2.3 do not depend on the electrode potentials, their rates decline as they compete with the

charge transfer reactions for reactants at the electrode interfaces. The net result is a shift in favor of the charge transfer reactions.

Power density is included in Figure 5.10 to show the relationship between coulombic efficiency and the desirable operating envelope of the fuel cell. Since cell potential *falls* with increasing power density, the coulombic efficiency *rises* with increasing power density. Gains in coulombic efficiency beyond the peak power point are undesirable, however, because they coincide with falling thermodynamic efficiency and are accompanied by greater total reactant fluxes to the electrodes. Beyond the peak power point, overall efficiency losses and greater reactant consumption rates offset any gains in coulombic efficiency, thus the maximum desirable coulombic efficiencies are those at peak power.

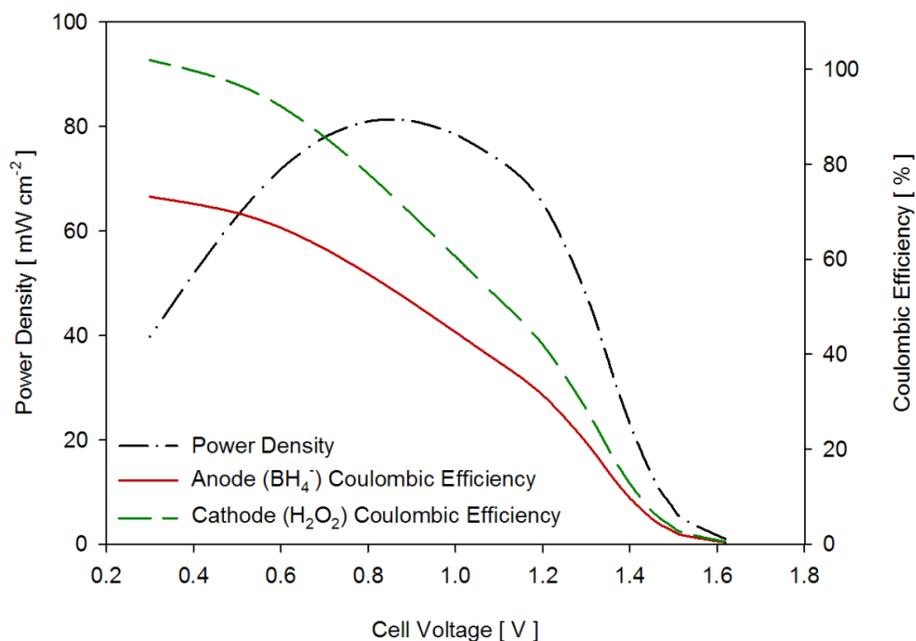


Figure 5.10. Predicted power density and coulombic efficiency for 50mM BH₄⁻ / 250mM H₂O₂ with 10 mL · min⁻¹ flow rates.

The ideal DBFC analysis in Chapter 3 showed that low single-pass fuel utilization is a challenge to DBFC practicality, and pointed to recirculation as a prospective solution. The extent to which the reactant solutions must be recirculated was evaluated in Chapter 3 with respect to the reactant utilization rates, assuming that all reactant consumption was due to charge transfer reactions. When competing reactions take place, the efficacy of recirculation is best described by the coulombic utilization η_{cu} , which is the fraction of reactant entering the cell that is consumed in a charge transfer reaction. The coulombic utilization was calculated as the ratio of the total cell current I to the current that would be available if all electro-active species k entering the cell were consumed in charge transfer reactions. Coulombic utilization is equivalently the product of reactant utilization η_{ru} and average coulombic efficiency over the whole channel (see Eq. 5.7).

$$\eta_{cu} = \frac{I}{C_k \dot{V} v_e / v_k} = \eta_{ru} \bar{\eta}_{ce} \quad \text{Eq. 5.7}$$

The coulombic utilization in the baseline case is plotted with respect to cell potential in Figure 5.11. The coulombic utilizations are reversed in comparison to the coulombic efficiencies; despite having a higher coulombic efficiency, the lower diffusivity of H_2O_2 leads to overall lower coulombic utilization in comparison to the anode. The slow rates of reactant transport through aqueous solution lead to low coulombic utilizations ($< 4\%$) for both sides of the cell under these operating conditions.

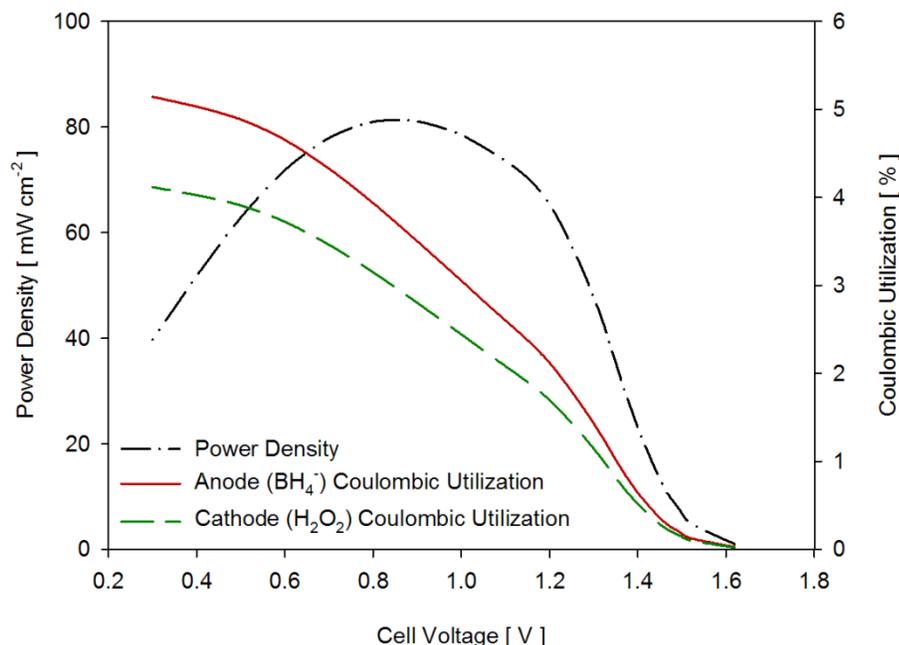


Figure 5.11. Plots of cell power density and coulombic utilization with respect to cell potential. Predictions for 50mM BH_4^- /250mM H_2O_2 with $10 \text{ mL} \cdot \text{min}^{-1}$ flow rates.

The losses and inefficiencies due to competing reactions are manifested in two ways: loss of reactants (and the chemical energy therein) to gas production, and lower cell potential via the Nernst equation (Eq. 1.8) as reactant concentrations near the electrodes are depressed. Cell potentials predicted by the calibrated model using the full reaction mechanisms are clearly lower than those predicted by the model with BH_4^- hydrolysis (reaction R 1.4) and H_2O_2 decomposition (R 1.5) turned off, as shown in Figure 5.12. Both curves in Figure 5.12 are plotted at $100 \text{ mA} \cdot \text{cm}^{-2}$ current density to make ohmic losses in the membrane comparable, with the two curves having the same electric potential gradients across the membrane. The power density in the full mechanism case is $80 \text{ mW} \cdot \text{cm}^{-2}$, whereas the power density in the case without hydrolysis or decomposition is nearly twice as high at $152 \text{ mW} \cdot \text{cm}^{-2}$. If the current density in the case without hydrolysis or decomposition were decreased to $0 \text{ mW} \cdot \text{cm}^{-2}$, the resulting electric potential profile would be the ideal case at OCV.

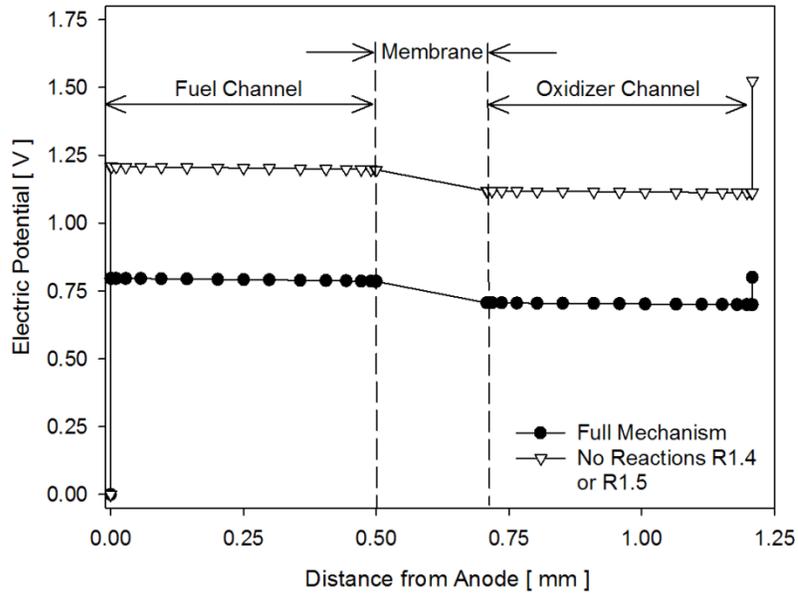


Figure 5.12. Electric potential profiles across the cell predicted by the calibrated model, at the midpoint (25 mm from the inlets), for the baseline case at $100 \text{ mA} \cdot \text{cm}^{-2}$ current density. The two curves compare results with R 1.4 and R 1.5 turned on and off.

One measure of the efficiency with which a DBFC extracts electrical energy from the reactants is the effective energy conversion efficiency. If the cell thermodynamic efficiency is expressed as the ratio of actual cell potential to theoretical OCV, then the effective energy conversion efficiency η_{eff} is given by Eq. 5.8.

$$\eta_{eff} = \eta_{ce} \frac{V_{cell}}{E_{R1.1}^0} \quad \text{Eq. 5.8}$$

The effective energy conversion efficiency shows a peak near the peak power point, indicating that (for this DBFC, in the baseline operating conditions) the most energy that can be extracted from a limited reactant supply by operating the cell at peak power. This is due to the opposing trends for thermodynamic efficiency and coulombic efficiency, and is in contrast to fuel cells which do not suffer from

parasitic side reactions, such as the PEMFC. The most efficient operating point for a PEMFC is near OCV, because thermodynamic efficiency is greatest near OCV.

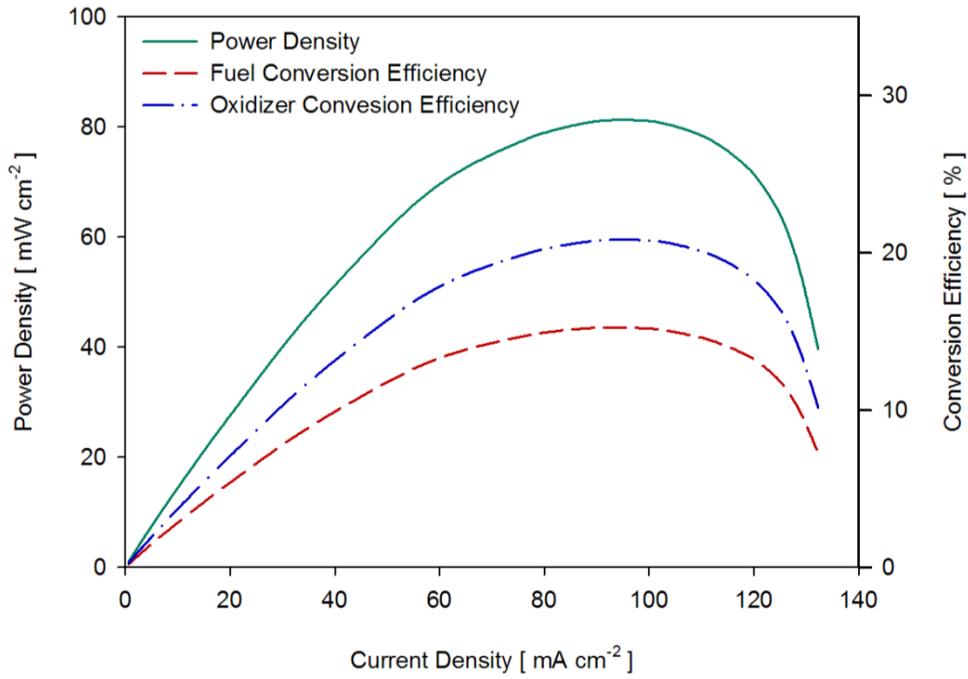


Figure 5.13. Effective conversion efficiencies for fuel and oxidizer in the baseline case, plotted with power density for comparison of the peak locations.

The overall peak efficiency operating point for a real PEMFC *system* is displaced from OCV by the minimum power consumed by the balance of plant (pumps, etc.), and the same should be true for a real DBFC. The overall peak efficiency for a DBFC system is likely shifted to lower net power output by the balance of plant load. For this reason, the DBFC model developed in this study was designed so that it could easily be subsumed into a larger system model for system-level analysis.

5.3.1 Influence of Fuel Flow Rate on Realistic Performance

The ideal DBFC analysis showed that raising the fuel or oxidizer flow rate improved performance by increasing the rates of convection mass transport in the channels. The same is true for the realistic DBFC, but with a caveat. The higher rates of reactant transport to the electrodes that come with higher flow rates also lead to higher rates of gas production. The influence of fuel flow rate on concentration boundary layer development is shown in Figure 5.14, where the BH_4^- concentration boundary layer at $10 \text{ mL}\cdot\text{min}^{-1}$ fuel flow rate is compact and the boundary layer at $1 \text{ mL}\cdot\text{min}^{-1}$ is beginning to envelop the entire channel. The compact boundary layer is the result of convection augmenting the rate of diffusion mass transport to the anode.

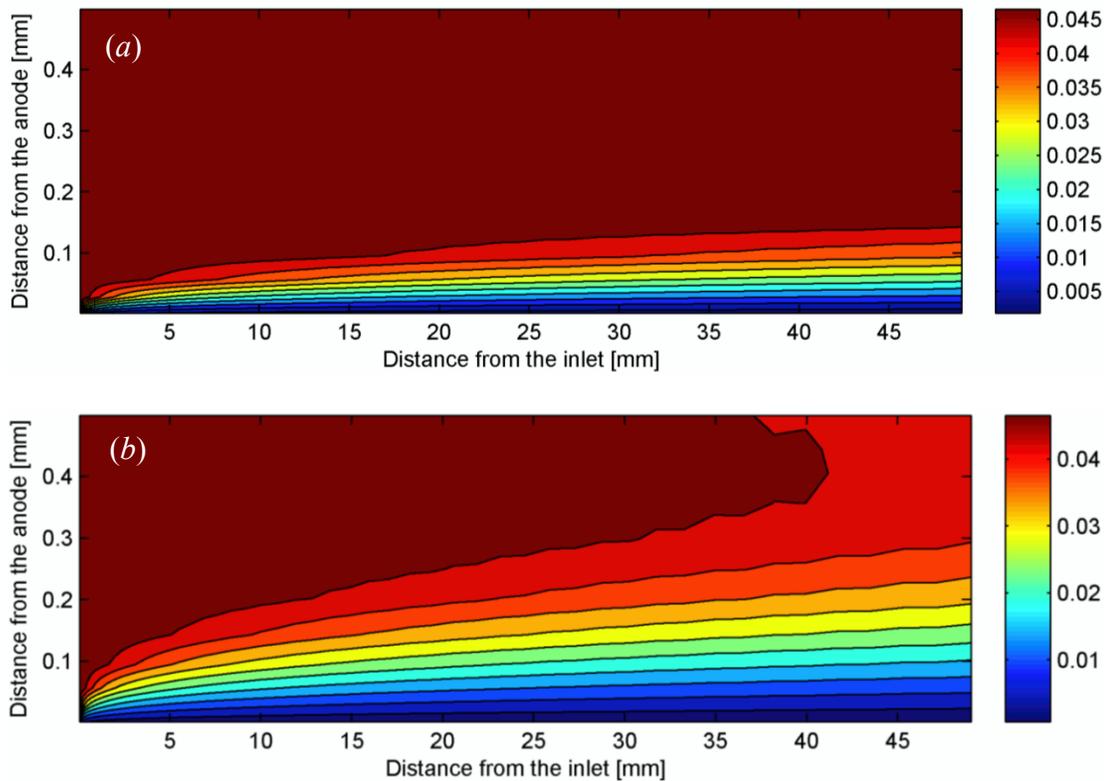


Figure 5.14. Predicted BH_4^- concentration in the fuel channel for $10 \text{ mL}\cdot\text{min}^{-1}$ (a) and $1 \text{ mL}\cdot\text{min}^{-1}$ (b) flow rates, for the baseline case at 1.1 V. Both plots share the same color map, with concentrations in $\text{mol}\cdot\text{L}^{-1}$.

The power density is plotted with respect to fuel flow rate in Figure 5.15, which shows the same trend of increasing peak power with flow rate that was found for the ideal case (Figure 3.13). The peak power point in the realistic DBFC does not shift to lower cell potentials with increasing flow rate as obviously as it did in the ideal analysis, however. The peak power point shifted in the ideal analysis because increasing fuel flow rates enabled higher current densities, which incurred greater ohmic losses in the membrane. In the realistic analysis, the current density at peak power density does not increase as quickly with fuel flow rate because coulombic efficiency and coulombic utilization at peak power density fall with increasing fuel flow rate, restraining the increase in current density. These trends are illustrated in Figure 5.16.

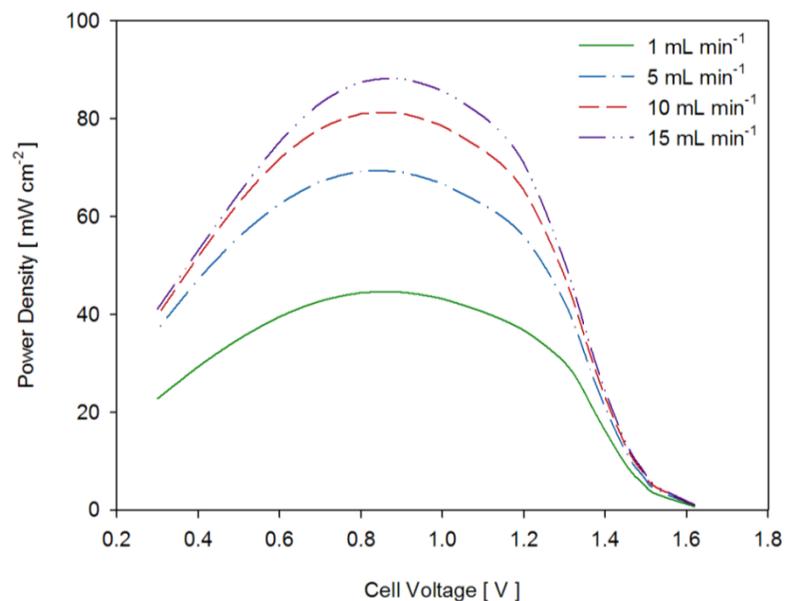


Figure 5.15. Predicted power density for fuel flow rates from 1-15 mL·min⁻¹, oxidizer flow rate of 10 mL·min⁻¹ and 50mM BH₄⁻/250mM H₂O₂.

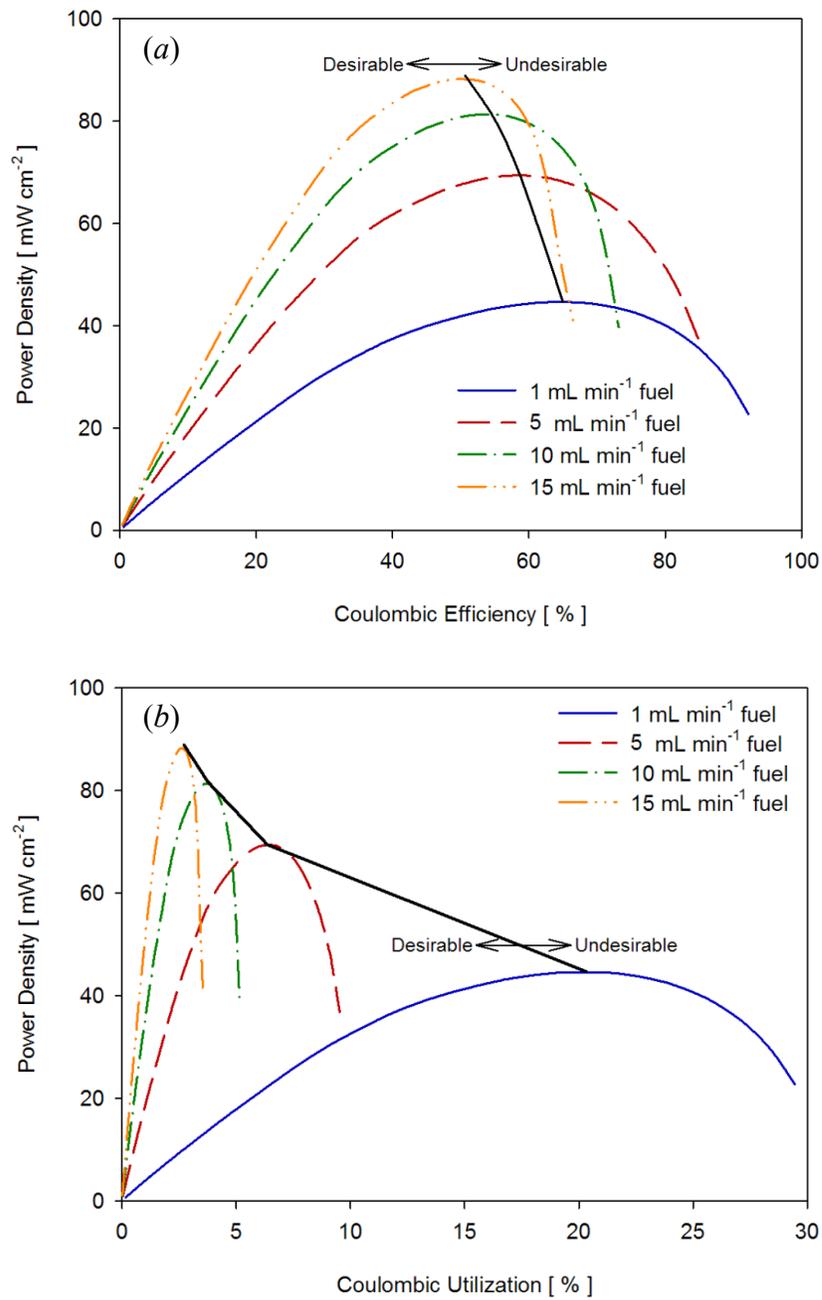


Figure 5.16. Predicted coulombic efficiency (a) and coulombic utilization (b) for fuel flow rates from 1-15 mL·min⁻¹, oxidizer flow rate of 10 mL·min⁻¹, and 50mM BH₄⁻ / 250mM H₂O₂.

The black line in Figure 5.16(a) traces the peaks of the power density curves, showing how greater power density is accompanied by a lower upper bound on the

coulombic efficiency. As the fuel flows more quickly, BH_4^- concentration at the anode interface rises in response to more facile transport from the bulk. The higher BH_4^- concentration favors hydrolysis, so the coulombic efficiency falls. The convex black curve approaches the horizontal axis as the fuel flow rate is lowered, until it intercepts the axis at zero flow rate. The maximum theoretical coulombic efficiency achievable for these operating conditions is $\sim 73\%$ and occurs in the zero flow rate case, in which diffusion and migration alone dictate the rates of transport to the anode. The power density shows a different relationship with coulombic utilization, with a concave black curve linking peak power points in Figure 5.16(b). The coulombic utilization at peak power increases as the flow rate and power density decreases.

5.3.2 Influence of Fuel BH_4^- Concentration on Realistic Performance

The influence of inlet concentration on realistic DBFC performance was examined with respect to the concentration of BH_4^- . Changing the inlet concentration revealed a relationship between power density and coulombic efficiency which mirrors that of the fuel flow rate (see Figure 5.17(a)). The similarity between the effects of fuel flow rate and BH_4^- concentration is unsurprising given that increases in either parameter shift the relative rates of reaction at the anode in favor of H_2 production by increasing BH_4^- concentration at the anode interface. The coulombic utilization trend with respect to BH_4^- concentration (see Figure 5.17(b)) differs from the trend with respect to flow rate. Slower flow rates increase residence times in the DBFC while diffusion and migration continue to transport BH_4^- to the anode, leading to the asymptotic approach to 100% utilization at zero peak power density seen in

Figure 5.15. Lower inlet concentrations decrease the rate of transport by all three processes (migration, diffusion and convection), leading to a theoretical upper bound of ~4.75% coulombic utilization at zero inlet concentration in Figure 5.17(b).

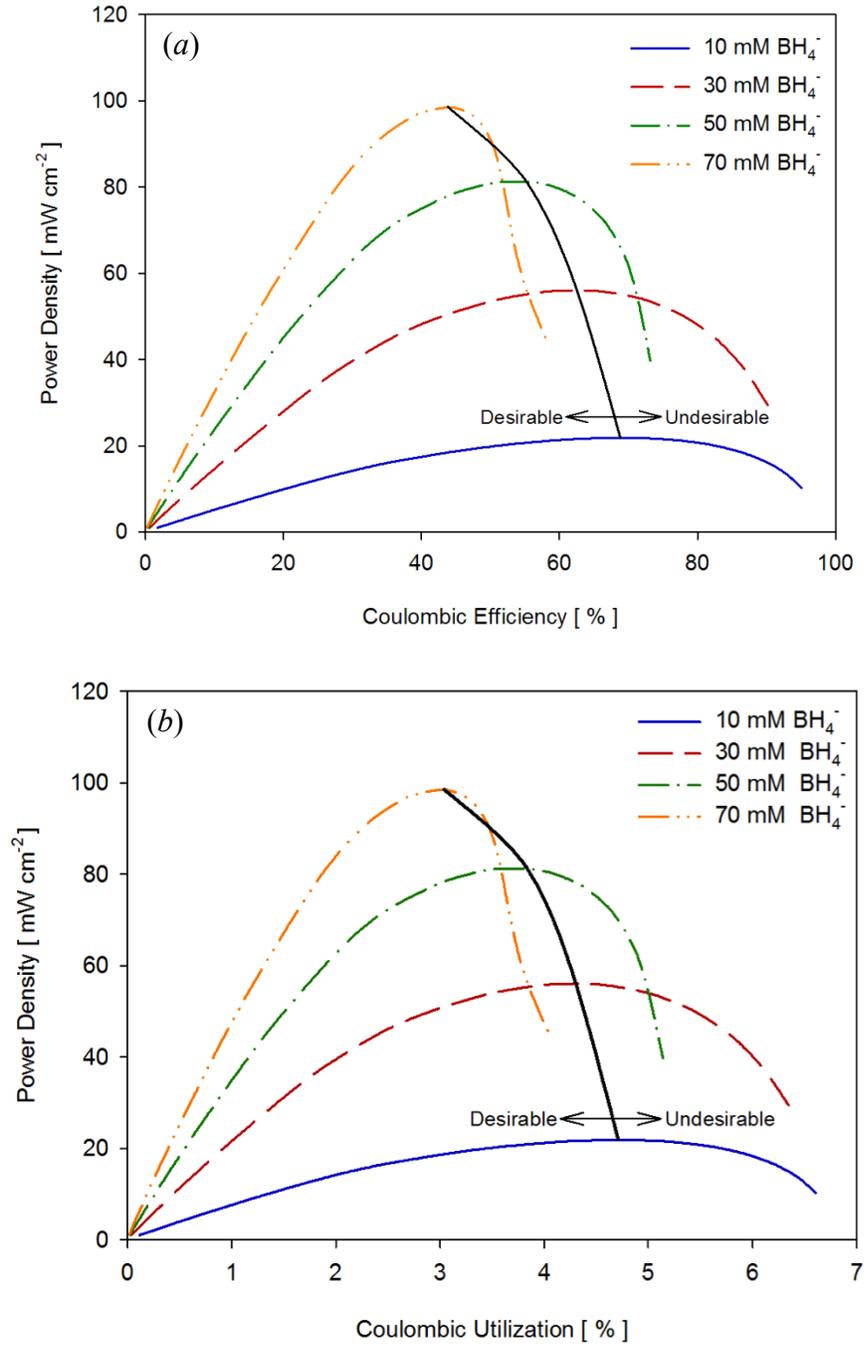


Figure 5.17. Predicted coulombic efficiency and utilization for BH_4^- concentrations from 10 – 70 mM, flow rates of $10 \text{ mL} \cdot \text{min}^{-1}$ and $250 \text{ mM H}_2\text{O}_2 / 1 \text{ M H}_2\text{SO}_4$ oxidizer.

5.3.3 Conclusions from the Realistic DBFC Analysis

The performance of a real DBFC is related to cell design and operating parameters in complex ways. Many trends are the same as in the ideal analysis, but with performance degraded by losses to BH_4^- hydrolysis and H_2O_2 decomposition. The trends are, in general, the same at the anode and cathode although the relative magnitudes differ. At the anode, for example, the peak power density increases with BH_4^- inlet concentration as it did in the ideal analysis, but differs in key ways.

With the side reactions included, power is delivered at lower cell potential due to competition with hydrolysis decreasing the BH_4^- concentration at the anode interface, which shifts the anode equilibrium potential according to the Nernst equation, increasing the activation overpotential necessary for a given current density. Furthermore, greater ion fluxes to/from the anode are required to support the combination of charge transfer and hydrolysis reactions, which incur greater ohmic losses in the fuel solution. Similarly, greater current density is required to achieve a given power density at the lower potential, which incurs greater ohmic losses in the membrane. Together, these effects decrease the thermodynamic efficiency of the cell.

Also with the side reactions, coulombic utilization is less than 100%, and substantially so for most operating conditions. Losses to competing reactions can cut the effective energy density of a system to half, or less, than the ideal case scenario. Recirculation still improves the overall coulombic utilization, but losses to hydrolysis will continue to consume a fraction of reactants with each pass through the cell.

Several promising strategies for mitigating the effects of hydrolysis and decomposition on the performance of DBFCs were revealed by the realistic DBFC analysis. Among these are operating at low cell potential and with low reactant concentration at the electrode interfaces; both strategies favor charge transfer reactions over hydrolysis and decomposition, shifting the relative reaction rates and improving the coulombic efficiency and utilization. There are several ways to achieve these favorable operating conditions, including lower reactant flow rates, lower inlet concentrations and higher recirculation fractions. Each change to the cell operating conditions has implications not only for the coulombic fuel utilization, but for the thermodynamic efficiency of the cell and the ability of a given system to satisfy design requirements for power output and total useful energy capacity. A more complete design trade analysis would examine the total effective conversion efficiency with respect to power density to choose a cell geometry and set of operating conditions that meet the requirements of a particular application.

Chapter 6: Conclusions

The four goals of this work outlined in Chapter 1 were all part of an underlying theme: applying cell-level models and experiments to the problem of understanding DBFC performance so that it can be improved. This approach has been applied to other fuel cell technologies with great success; the PEMFC is a prominent example. DBFC technology is still in its infancy compared to more mature technologies like the PEMFC. This relative immaturity has two implications for cell-level DBFC modeling and experimental studies. First, the small body of prior work provides an open field for substantially improving our understanding of DBFC performance. Second, modeling and experimental tools remain underdeveloped. The purpose of this study was to lay some of the groundwork for modeling DBFCs (Chapter 2), show how the models can be used to analyze design trends (Chapter 3), and show how the interplay of DBFC modeling and cell experiments could provide improved understanding of DBFC performance and guide design decisions (Chapters 4 and 5).

6.1 Factors Governing DBFC Performance

The combination of modeling and experiments in this study yielded a greater understanding of DBFC performance, and how it might be improved. Broadly speaking, the parameters dictating DBFC fall into two categories: transport and electrode reaction related phenomena.

6.1.1 Influence of Transport on Performance

As the modeling results of Chapters 3 and 5 showed, rapid development of compact concentration boundary layers in both channels make power density depend heavily on inlet reactant concentration. The compact nature of the concentration boundary layers and slow growth rate at points far from the inlet may be due, in part, to migration contributing to transport in the bulk. The slow broadening of the concentration boundary layers makes the average power density (for a given set of operating conditions) relatively insensitive to channel length. Peak power output can be increased by higher reactant concentrations and/or solution flow rate, because these boundary layers factor so prominently in determining peak power.

All three transport processes (diffusion, migration and advection) contribute to the transport of reactants and products in the channels. Diffusion and advection (together, convection) dominate transport near the electrodes. In the bulk, migration of Na^+ is crucial for charge balancing the cell and in general, reactant transport is dominated by migration and advection. Advection fluxes in the y -direction due to water crossing through the membrane can be significant, and the interaction of this advection flux with a membrane impermeable to some species can influence the concentrations of all species near the membrane due to the electroneutrality condition. Specifically, the y -direction advection flux coupled with a membrane impermeable to anions will lead to depressed anion concentrations near the membrane, which depress the cation concentrations as well. In the ideal case analysis, the cell incurred a small (1%) efficiency loss due to these concentration gradients.

Ohmic losses in the channels are small compared to losses in the membrane, where the lower mobility of Na^+ ions limits conductivity compared other cations, such as H^+ . The membrane ohmic overpotential was small in this study compared to the activation overpotentials at the electrodes (particularly so for the larger cathode activation overpotential), but it was still a significant source of loss. For example, the baseline case of the realistic DBFC analysis at 1.1 V cell potential showed the membrane ohmic losses were 3.5% of total potential loss compared to the theoretical 3.01 V cell potential. The importance of membrane ohmic loss increases with current density, so it would become more significant in systems using higher reactant concentrations and/or higher flow rates to achieve higher current density. Channel ohmic losses are negligible for any reasonable combination of reactant and supporting electrolyte concentrations (≤ 2 M) and channel depth (≤ 1 mm). Also, for reasonable concentrations of OH^- and H^+ , crossover of these species through the membrane is negligible at cell potentials less than OCV. With higher concentrations, crossover of OH^- and H^+ may occur at open circuit and depress OCV, but remains unlikely to have a significant impact on cell performance at lower cell potentials, for which the electric potential gradient in the membrane opposes crossover.

Reactant stoichiometry was shown to affect DBFC behavior by altering the relative rates at which the anode and cathode potentials change with increasing current density. Fuel (BH_4^-) limited cases showed consistent trends of increasing current density with increasing BH_4^- concentration as the higher concentration drove larger BH_4^- fluxes to the anode. The limiting current density in these cases was roughly proportional to the BH_4^- concentration. Oxidizer (H_2O_2) limited cases

showed that shifting most of the concentration overpotential to the cathode could cause its potential to fall far enough for additional charge transfer reactions (H^+ reduction) to become thermodynamically favorable. The onset of additional charge transfer reactions showed that changes to stoichiometry could change cell behavior at high current density in ways which deviate from the proportionality dictated by convection transport.

6.1.2 Influence of Electrode Reactions on Performance

Electrode reaction rates influence DBFC performance in two ways. First, the activation and concentration (manifested through the activation) overpotentials are the largest factors determining the relationship between the cell potential and current density. In the baseline case of the realistic DBFC analysis at 1.1 V cell potential, the total anode and cathode overpotentials ($\eta_{act} + \eta_{conc}$) constituted 20% and 75% of total potential losses, respectively. Second, the coulombic efficiency is determined by the relative rates of the desired charge transfer reactions and competing chemical reactions. These relative reaction rates also play a role in the coulombic utilization.

The rate of H_2 production at the anode tends to be greater than the rate of O_2 production at the cathode, yet H_2 production has less effect on the anode overpotential than O_2 does on the cathode. The reasons involve transport and stoichiometry. BH_4^- has higher diffusivity than H_2O_2 and is augmented by migration, so the concentration overpotential at the anode tends to be lower. Furthermore, each BH_4^- anion to arrive at the anode can yield up to $8 e^-$, so the coulombic efficiency at the anode can fall to 25% and still balance the cathode current density for a given molar consumption flux

at both electrodes. Even a small decrease in the cathode coulombic efficiency forces a large increase in the cathode overpotential to match the current density of the anode.

At the anode, Au does not favor complete BH_4^- oxidation as claimed in many publications, although the rate of H_2 production may be less than that of more active catalysts such as Pt. Qualitative observations in *ex situ* experiments showed that the rate of H_2 production increased with anode potential, likely due to higher rates of anion adsorption as the anode became less negative. Despite the higher net rate of H_2 production, model results showed that the relative rates of BH_4^- oxidation and H_2 production favored BH_4^- oxidation (higher coulombic efficiency) at less negative anode potentials.

At the cathode, large overpotentials cause the electrode potential to fall rapidly with increasing current density, and it can reach potentials sufficiently low for H^+ reduction to occur. The onset of a second charge transfer reaction at low potential can delay the transport limit to lower cell potentials and increase the peak power density of the cell. In general, however, most H^+ reduction occurs at points on the power density curve which are undesirable from an efficiency perspective. Bubble production was observed at open circuit (O_2) and at low cathode potential (H_2) in cell and *ex situ* experiments. Few bubbles were observed at intermediate cathode potentials.

6.1.3 Recommendations for Improved DBFC Performance

In terms of cell design and operation, long shallow channels are preferable to improve both coulombic utilization and efficiency. As demonstrated by Figure 5.13, the highest overall conversion efficiency is obtained close to peak power density for

this DBFC, rather than near OCV as is common for many fuel cells. Flow rate and inlet reactant concentrations can be adjusted to maintain operation near peak power as the load varies. If reactant storage volume is limited, the total energy capacity of the system may be greatest with a low fuel:oxidizer storage ratio, so that the cell can operate with more excess oxidizer to mitigate activation and concentration losses at the cathode and raise the overall conversion efficiency. Similarly, operating the cell with an oxidizer flow rate greater than the fuel flow rate to compensate for slower transport of H_2O_2 may be a good compromise between cell thermodynamic efficiency and parasitic power consumption by the recirculation pumps.

Due to the low reactant utilization rates, even for channels as thin as 0.5 mm deep, recirculation will be necessary for any practical DBFC. With recirculation, some fluid must be rejected downstream of the cell to accommodate the addition of new reactants; the rate of reactant loss to this process will be mitigated by operating the cell with low average reactant concentrations.

The high rate of water crossover through the membrane introduces an important caveat to the recommendation to run near peak power. Water crosses the membrane predominantly due to electro-osmotic drag, making the crossover rate proportional to the current density. Depending on the system topology and design constraints for a given application, it may be preferable to operate the cell close to OCV in order to produce sufficient power with the smallest possible current density, thereby minimizing the water crossover rate.

The present study analyzed a cell with catalyst layers on planar electrodes separated from the membrane. This topology was chosen because it was

straightforward to model and because of advantages such as simplicity of design, few processing steps in the fabrication process, and migration aiding reactant transport. This cell topology is not necessarily the best for all applications, however. Cells in which the flow channels are filled with a porous solid catalytic medium may offer better performance by removing concentration boundary layer development as an impediment to higher current density (power density) operation. Concentration gradients would still develop, but forcing the reactant solutions through a porous medium would improve rates of transport from the bulk to the surface by distributing those gradients over a larger surface area. Furthermore, the larger catalytic surface area could lessen the need for recirculation by improving coulombic utilization rates.

6.2 *Prospects for Practical DBFCs*

The high theoretical energy density and air independence of $\text{NaBH}_4 / \text{H}_2\text{O}_2$ DBFCs continue to make this technology attractive for portable power and undersea applications, yet the technology remains too immature to be practical. The greatest obstacles to implementation continue to be losses to competing electrode reactions, water crossover through the membrane, ohmic losses in the membrane and activation losses at the cathode. The strategies for mitigating these problems recommended by this study can help to improve DBFC performance, but more progress must be made before DBFCs can be practical. The modeling tools developed in this study should be useful for investigating possible solutions, such as catalysts with higher activity and selectivity, membranes with higher conductivity and system configurations which minimize water crossover.

6.3 *The Utility and Limits of (Present) DBFC Models*

The ability of this model to predict performance characteristics such as power density and efficiency makes it a valuable tool for system design. It can be used in several ways, including:

- (a) Optimizing for one or more steady-state performance metrics in a given system by choosing appropriate operating conditions.
- (b) Providing insight into difficult to measure system parameters, such as the rate of hydrolysis down the channel, to inform cell design decisions.
- (c) Guiding development of control strategies which optimize one or more performance metrics over a range of operating conditions.

If the model is sufficiently accurate, it can be used to develop a reduced state estimator for a model predictive control strategy. Such an approach may be well suited to DBFCs because the complex relationships between operating conditions and performance complicate the selection of operating conditions, and because the variables of greatest interest for control (such as coulombic utilization) can be difficult to measure in real time.

This study showed that models must accommodate down-the-channel variations in transport and reaction rates in order to accurately predict cell performance. The finite volume approach worked well for this system and provided a flexible and robust means to solving the transport equations for the channel and membrane. The generalized reaction rate estimation functions readily accommodated both chemical and charge transfer reactions when the appropriate rate parameters were provided.

The largest challenge to DBFC model development is identifying electrode reactions with sufficient detail to adequately describe the system under study. The global reaction rates fitted to the measurements in this study were adequate, but discrepancies remained. To produce high fidelity performance predictions for system design, it may be necessary to incorporate more complex reaction rate models. A framework for incorporating surface species and adsorption reactions was laid out in Chapter 2, but was not adopted here in part because the rate parameters for such a mechanism are not readily available. While progress has been made, measurement of rate parameters for such a complex microkinetic rate mechanism continues to be a challenge for reactions as complex as BH_4^- oxidation.

6.4 *Products of this Study*

The products of this study include modeling tools for DBFC analysis and calibration, calibrated reaction rate mechanisms for Au and Pd:Ir, experimental results from a DBFC with geometry that is amenable to modeling, and experimental data which can be used to calibrate future iterations of this model or other models. Beyond these concrete products, however, is an improved understanding of the factors dictating DBFC performance.

In terms of disseminating the conclusions of this study, one paper describing the model development and ideal DBFC analysis was accepted to the Journal of Power sources [95]. Talks on this work were delivered at three conferences: the 2012 Fuel Cell Seminar, the 2013 ASME Fuel Cell Science and Technology Conference and the

2013 NRL Chemistry Division Symposium. A second paper describing the experiments, model calibration and realistic DBFC analysis is in preparation.

6.5 *Recommendations for Future Work*

Future work could take several paths. The present model could be modified to include an anode reaction mechanism involving adsorption and surface species to better capture the shift in anode reaction rates with electrode potential, and the possibility of intermediates escaping prior to complete oxidation. Such a more detailed mechanism will be feasible if the relevant rate parameters become available. High concentration reactants could be investigated to understand whether or not the underlying dynamics governing DBFC performance change when the electrolyte solutions become strongly non-ideal. Such a high concentration investigation would have to adopt a method for estimating activity coefficients, with the Pitzer equations being the best choice if the necessary species properties are available.

The model could be modified to accommodate multiphase flows and bubble nucleation at the catalyst layers. Adding these details could reveal the relationships between gas production rate and performance degradation when bubbles occlude the electro-catalysts. The transport features of the model could be modified to fill the channels with a catalytic porous medium and compare the performance of that cell topology to the cell examined in this study.

As discussed in Chapter 5, the model was designed to fit into a system-level DBFC model. A major feature of such a model would be recirculating flows, which could be established by assigning the outlet flow conditions (concentrations and flow

rate) to the inlet boundary conditions, with appropriate adjustments to simulate waste removal and reactant injection. This study examined steady state DBFC performance, which was reasonable given the relatively slow dynamics of the cell. A transient system-level analysis may be desirable to investigate cell interactions with varying operating conditions such as flow rate and inlet concentration. The present model is well suited to this purpose, because the governing equations were written such that the residuals are time derivatives of the associated state variables. The model could straightforwardly be adapted to work with a time integrating solver for transient analysis.

Chapter 7: Appendices

7.1 *DBFC Model Code*

```
%% SUMMARY: DBFC_MAIN
% Purpose: Call the scripts and functions to set up the model, solve it and display the results. Author:
% Rick Stroman

%% NOTES
%
% To change the way the code runs, modify the global variables Geometry, Model, Flags, Scales, Fuel and
% Oxidizer in the script DBFC_USER_INPUT. This is the ONLY place these global variables are created
% and/or changed... elsewhere they are ONLY READ, NOT ALTERED. Nothing else should be changed from one run
% to another, unless you want to change the model.

%% CODING CONVENTIONS
%
% (1) Script and function names are in "ALL_CAPS"
%
% (2) Vectors, structures and matrices have first letter "Capitalized"
%
% (3) Scalars are all lower "case"
%
% (4) Top level functions and script names begin with DBFC_ , sub-functions begin with FUNC_ and
% sub-scripts begin with SCRIPT_ .
%
% (5) Major sections of code are broken into MATLAB cells and labeled with comments in all caps, and
% subsections have comment headings with ordinary text.
%
% (6) Quantities restricted to one discretization start are stored in structures beginning with the
% discretization name; for example, Asln.mass_density. Fluxes among discretizations are stored in
% structures starting with the flux direction, followed by the flux type and flux name. For example,
```

```

% Yfluxes.mass.asln is the y-direction mass flux leaving discretization asln. Numerical indexes
% following either type of variable are ordered (x-discretization, y-discretization, species number).
%
% (7) Abbreviations and acronyms
%
% DCS: Discretization Coordinate System. This includes ghost cells for boundary conditions and is
% denoted x_d and y_d. x_d = 2, y_r = 2 is the corner real cell adjacent to the inlet and
% electrode. x_r = Geometry.x_d_num + 1 and y_r = Geometry.y_d_num + 1 is the corner real cell
% adjacent to the outlet and membrane.
%
% RCS: Real Coordinate System. This does not include ghost cells and is denoted x_r and y_r. x_r =
% 1, y_r = 1 is the corner cell adjacent to the inlet and electrode. x_r = Geometry.x_d_num and
% y_r = Geometry.y_d_num is the corner cell adjacent to the outlet and membrane.
%
% DBFC: Direct Borohydride Fuel Cell

%% 1. SET UP THE WORKSPACE

close all; clc; clear all;

global BC Scales Geometry Pointer Solver total_cathode_current Grid_size

%% 2. SET UP THE MODEL

% Set user configurable options such as cell geometry, model properties, solver parameters, etc. This is
% the only place where model functionality is changed from one run to the next.
DBFC_USER_INPUT;

% Set up the MATLAB workspace
DBFC_CONFIGURE_WORKSPACE

% Define physical constants and species properties.
DBFC_CONSTANTS_AND_PROPERTIES;

% Calculate additional parameters from the user input, outside of the solver to be more efficient.
DBFC_PROCESS_USER_INPUT

```

```

% List the reactions included, with stoichiometric and rate parameters for each. Store all of the
% stoichiometric and rate parameters in structures expected by the reaction rate function.
DBFC_SETUP_REACTION_RATES

% Create pointers, names, scales and constraints vectors. Generate a Jacobian pattern, mass matrix and
% populate the initial state vector.
DBFC_INITIALIZE

%% 3. SOLVE THE MODEL

voltage_num_tot = length(Cathode_electric_potential);

% Solve model for each cell voltage specified in Cathode_electric_potential
for voltage_index = 1:voltage_num_tot

    % SET THE CELL VOLTAGE AND CALL THE REQUESTED SOLVER TO SOLVE THE MODEL
    BC.cathode.elec_pot = Cathode_electric_potential(voltage_index);

    % CALL THE SOLVER TO FIND STEADY STATE SOLUTION AT THIS CELL VOLTAGE
    DBFC_KINSOL

    % IF GENERATING A POLARIZATION CURVE, RUN THE REST OF THE VOLTAGES

    if voltage_num_tot > 1
        % If we are running more than one cell voltage, do this stuff after the first one...

        % Store the current density from this voltage in an array so we can plot the pol curve later
        current_density(voltage_index) = -total_cathode_current / (sum(Geometry.y_flux_area));
        anode_delta_phi(voltage_index) = 0 - mean(SV_steady_state(Pointer.a_int.elec_pot));
        cathode_delta_phi(voltage_index) = Cathode_electric_potential(voltage_index) ...
            - mean(SV_steady_state(Pointer.c_int.elec_pot));

        DBFC_RESULTS_OUTPUT

        total_BH4_consumed(voltage_index) = total_BH4_to_anode;
    end
end

```

```

total_H2O2_consumed(voltage_index) = total_H2O2_to_cathode;

% Display status in the command window
disp(' ')
disp(['Point number ', num2str(voltage_index), ' of ', num2str(voltage_num_tot), ' is complete.'])
disp(['Cell voltage is: ' num2str(BC.cathode.elec_pot), ' V'])
disp(['Total cell current is: ' num2str(total_cathode_current) ' A'])
disp(['Average cell current density is: ' num2str(current_density(voltage_index)) ' A/m^2'])

% SAVE WORKSPACE TO A .mat FILE
if Flags.setup.save_inter_output
    % If saving the workspace after each cell voltage:
    c = clock;
    filename_id = strcat(num2str(c(1)), '-', num2str(c(2)), '-', num2str(c(3)), '-', ...
        num2str(c(4)), 'h-', num2str(c(5)), 'm-', num2str(c(6)), 's', '_', ...
        num2str(BC.cathode.elec_pot), 'V');
    filename = strcat('Inter_Results_', filename_id, '.mat');
    save(filename); disp(['Results were saved in file: ' filename])
    clear c filename_id filename
end

if Flags.setup.reuse_previous_soln && ...
    voltage_index < voltage_num_tot

    % Change the strategy to be pure Newton search, in case LineSearch was used to get the simulation
    % started
    Solver.kinsol.strategy = 'None';

    % Configure the solution from the previous voltage to be the initial guess for the next voltage
    if Flags.setup.adjust_V_prev_soln
        % Adjust the solution from the previous voltage to form the initial guess for the next point
        V_cell_old = Cathode_electric_potential(voltage_index);
        V_cell_new = Cathode_electric_potential(voltage_index+1);
        SV_initial = FUNC_ADJUST_SOLN_VOLTAGE( SV_steady_state, V_cell_old, V_cell_new, Scales, ...
            Pointer, Geometry, Grid_size );
    else

```

```

        % Use the solution from the previous voltage as-is
        SV_initial = SV_steady_state;
    end
end

end

% Loop to walk down the polarization curve, one cell voltage at a time
end

%% 4. SUMMARIZE SIMULATION

% Stop the clock and display end message.
elapsed_time = toc; disp(' ');
disp(['Simulation complete. Total time for solver to run was ' num2str(elapsed_time) ' s.' ])

%% 5. ANALYZE AND DISPLAY THE RESULTS

% Plots output for a single solution
if voltage_index == 1
    DBFC_RESULTS_OUTPUT
end

if voltage_index > 1 && ~pol_curve_compare
    figure(1);
    plot( 0.1*current_density, Cathode_electric_potential, 'r-o')
    title('Calculated Polarization Curve'); xlabel('Current Density [mA/cm^2]'); ylabel('Cell Voltage [V]')
end

if voltage_index > 1 && pol_curve_compare
    figure(1);

    measured_current_densities = [ 0.00; 0.62; 0.92; 1.63; 2.77; 5.26; 9.67; 16.63; 47.00; 69.93; ...
        81.56; 90.84; 100.73; 109.67; 118.40; 126.42; 128.87; 131.34 ]; % Exp54H in mA/cm^2

    figure(1);

```

```

plot( 0.1*current_density, Cathode_electric_potential, 'r-o', measured_current_densities, ...
      Cathode_electric_potential, 'b-s')
title('Calculated and Measured Polarization Curves'); xlabel('Current Density [mA/cm^2]');
ylabel('Cell Voltage [V]')
legend('Calculated', 'Measured')

figure(2);
plot( 0.1*current_density, anode_delta_phi, 'b-o', 0.1*current_density, cathode_delta_phi, 'r-o', ...
      0.1*current_density, Cathode_electric_potential, 'g-o')
title('Electrode Potentials wrt Interface'); xlabel('Current Density [mA/cm^2]');
ylabel('Potential [V]')
legend('Anode', 'Cathode' , 'Cell')

figure(3);
plot( Cathode_electric_potential, anode_delta_phi, 'b-o', Cathode_electric_potential, ...
      cathode_delta_phi, 'r-o')
title('Electrode Potentials wrt Interface'); xlabel('Cell Voltage [V]'); ylabel('Potential [V]')
legend('Anode', 'Cathode')

end

if Flags.setup.save_final_output
    c = clock;
    filename_identifier = strcat(num2str(c(1)), '-', num2str(c(2)), '-', num2str(c(3)), '-', ...
        num2str(c(4)), 'h-', num2str(c(5)), 'm-', num2str(c(6)), 's');
    filename = strcat('Results_', filename_identifier, '.mat');
    save(filename); disp(['Results were saved in file: ' filename])
end
% Purpose: Set user defined model parameters such as cell geometry, voltage, etc. Author: Rick Stroman

%% DECLARE MODEL-WIDE GLOBAL VARIABLES

global Flags Geometry Pointer BC Initial Tolerances Solver Grid_size Species

disp('Reading user inputs...');

```

```

%% 1. MODEL OPTIONS

% SOLUTION APPROACH

Flags.setup.parallel = 0;
% [ 1 = run in parallel, 0 = run serially ]
Flags.setup.MEX      = 1;
% [ 1 = run MEX file to calc properties and fluxes, 0 = run MATLAB code ]

% SOLUTION AND RESIDUAL SCALES

Flags.setup.rescale_init_guess = 0;
% [ 0 = Leave the initial guess as-is, 1 = Adjust Scales.SV so that the initial guess is all ones ]

Flags.setup.rescale_init_resid = 0;
% [ 0 = Leave the initial residuals as-is, 1 = Adjust Scales.dSV so the initial residuals are all ones ]
% SOURCES FOR THE INITIAL GUESS

Flags.setup.new_SV_initial = 1;
% [ 1 = Initialize to the standard guess, 0 = Initialize to a guess from a solution file
% NOTE: This flag determines which guess is initialized. If Flags.setup.reuse_SV_steady_state = 0 below,
% then it is also used for subsequent points]

Flags.setup.reuse_previous_soln = 1;
% [ 0 = Use the guess from the Initialization for all voltages, 1 = Use the guess from the Initialization
% for the first voltage,
% then reuse solutions for subsequent voltages. ]

Flags.setup.adjust_V_prev_soln = 0;
% [ 0 = Use the previous solution as-is for the initial guess, 1 = Adjust the voltages in the previous
% solution to be closer to
% the voltages in the case for which it is the initial guess. NOTE: Only applicable when using a
% previous solution, either from
% file or from a previous voltage in a pol curve ]

Flags.setup.randomize_SV = 0;

```

```

% MODEL CONFIGURATION

Flags.model.scale_cells_x = 'lin';
% ['off' or 'lin' or 'log' for logarithmic, must use even numbers of cells ... makes cells smaller near
% inlets and walls ]
Flags.model.scale_cells_y = 'lin';
% ['off' or 'lin' or 'log' for logarithmic, must use even numbers of cells ... makes cells smaller near
% inlets and walls ]
Flags.model.solution_ideality = 1 ;
% [ 1 = ideal, 0 = non-ideal ]
Flags.model.y.migration = 1;
% [1 : include y-direction migration in transport calculations in the electrolyte]
Flags.model.y.diffusion = 1;
% [1 : include y-direction diffusion in transport calculations in the electrolyte]
Flags.model.x.migration = 1;
% [1 : include x-direction migration in transport calculations in the electrolyte]
Flags.model.x.diffusion = 1;
% [1 : include y-direction diffusion in transport calculations in the electrolyte]
Flags.model.m.migration = 1;
% [1 : include migration in membrane transport]
Flags.model.m.diffusion = 1;
% [1 : include diffusion in membrane transport]
Flags.model.m.permeation = 1;
% [1 : include permeation in membrane transport]
Flags.model.m.EOD = 1;
% [1 : include electro-osmotic drag (of water) in membrane transport]
Flags.model electroneutrality = 1;
% [1 : solve for electric potential in channel using electroneutrality
% 0 : solve for electric potential in channel using poisson electrostatic equation]

% KINSOL SETUP

% kinsol options
Solver.kinsol.display_iter = 1;
Solver.kinsol.verbose = false;

```

```

Solver.kinsol.func_norm_tol      = 1e-3;
% Stopping tolerance on the residual 2-norm
Solver.kinsol.scaled_step_tol    = 1e-15;
% Stopping tolerance (minimum) step size
Solver.kinsol.linear_solver      = 'Band';
% 'Dense' 'Band' or 'GMRES' 'TFQMR' 'BiCGStab'
% Solver.kinsol.KrylovMaxDim      = 10; Solver.kinsol.MaxNumRestarts    = 2;
Solver.kinsol.MaxNumSetups       = 50;
Solver.kinsol.strategy           = 'None'; % 'LineSearch' or 'None'
Solver.kinsol.MaxNewtonStep      = 1e9; % Default is 1e3
Solver.kinsol.MaxNumBetaFails    = 50; % Default is 10
Solver.kinsol.MaxNumIter         = 500; % Default is 200

% Jacobian options
Solver.Jacobian.Jpattern_flag    = 'none';
% [ 'random', 'load', 'analytic', 'ones', 'empirical' 'specified' or 'none' ]
Solver.Jacobian.JAC_bandwidth_flag = 'user';
% [ 'user' = user supplied upper and lower Jabobian bandwidths 'Jpattern' = determine from Jacobian pat]
Solver.Jacobian.rtrn_dense_JAC_flag = 0; % [ 0 or 1 ]
Solver.Jacobian.function         = 'FUNC_JACOBIAN_NUMJAC';
% External function for calculating the Jacobian, if that option is used.
Solver.Jacobian.upper_bandwidth  = 359;
Solver.Jacobian.lower_bandwidth  = 359;

%% 2. FUNCTION SCALES

% Scales for the solution vector... initially divide SV by these factors to get a value of order 1.
% Converting back to the real value inside the function is a multiplication, which is fast.
Solver.SV_scale.mass_frac_anode  = [ 1e-3 1e-5 1e-3 1e-1 1e-2 1e-2 ]; % Fuel.Mass_fractions
Solver.SV_scale.mass_frac_cathode = [ 1e-3 1e-6 1e-1 1e-3 1e-5 1e-3 1e-1 ]; % Oxidizer.Mass_fractions
Solver.SV_scale.x_vel            = 1e-2;
Solver.SV_scale.y_vel            = 1e-4;
Solver.SV_scale.press             = 1;
Solver.SV_scale.elec_pot         = 1;

% Scales for the residuals... these are what we multiply by inside the function to get values which have

```

```

% a similar magnitude near the solution.
Solver.dSV_scale.electroneutrality      = 1e1*1e0;
Solver.dSV_scale.species_cons_a        = 1e4 * [ 1e1 1e1 1e1 1e0 1e1 1e1 ];
Solver.dSV_scale.species_cons_c        = 1e4 * [ 1e1 1e1 1e0 1e1 1e1 1e1 1e1 ];
Solver.dSV_scale.x_momentum_cons       = 1e0;
Solver.dSV_scale.y_momentum_cons       = 1e0;
Solver.dSV_scale.mass_cons              = 1e0;
Solver.dSV_scale.species_flux_balance_a = 1e4 * [ 1e1 1e1 1e1 1e0 1e1 1e1 ];
Solver.dSV_scale.species_flux_balance_c = 1e4 * [ 1e1 1e1 1e0 1e1 1e1 1e1 1e1 ];
Solver.dSV_scale.mass_flux_balance      = 1e0;

%% 3. PLOTTING AND DISPLAY OPTIONS

Flags.setup.display_initial_state = 0;
% [1 = display initial state of system before starting the solver]
Flags.setup.display_final_state   = 0;
% [1 = display final state of system after the steady state solver is finished]
Flags.setup.display_aspect_ratios = 0;
% [1 = display aspect ratios of channel discretizations]
Flags.setup.plot_curr_density     = 0;
% [1 = display current density down the channel for anode and cathode at each iteration. 0 = don't
% display the current density at each iteration]

Flags.setup.display_Jpattern = 0; % [1: display the Jacobian pattern]
Flags.plots.grids            = 0; % [1: plot the channel grids]

% If this is a steady state simulation, plot the following:
Flags.plots.anode           = 0;
Flags.plots.cathode         = 0;
Flags.plots.whole_cell      = 1;

Flags.plots.contour         = 0;
Flags.plots.profiles        = 0;
Flags.plots.image           = 0;
Flags.plots.yfluxes_mass    = 0;
Flags.plots.yfluxes_mole    = 0;

```

```

Flags.plots.xfluxes_mass = 0;
Flags.plots.xfluxes_mole = 0;

Flags.plots.error      = 0;

pol_curve_compare = 1;
display_output_text = 1;

%% 4. READING AND WRITING FILES

% If using a previously stored Jacobian pattern, load it from the following file.
if strcmp(Solver.Jacobian.Jpattern_flag, 'load')
    Solver.Jacobian.j_pattern_filename = 'Jacobian_pattern_x60_y_15.mat';
end

% If using a previously stored solution vector, load it from the following file.
if ~Flags.setup.new_SV_initial
    Solver.SV_initial_filename = 'Inter_Results_2013-10-22-16h-48m-57.296s_1.6197V.mat';

    % File at voltage previous to the initial filename. Used to calculate the rate at which state
    % variables change with respect to cell voltage, to generate a better initial guess than just starting
    % at the last good solution.
    Solver.SV_previous_filename = 'Results_2013-8-1-9h-43m-58.491s.mat';
end

% Configure whether or not the calculated Jacobian and Jacobian pattern are saved to files
Flags.setup.save_Jacobian = 1;
Flags.setup.save_JAC_pat  = 1;

Flags.setup.save_final_output = 1;
% [1 = save the final workspace]
Flags.setup.save_inter_output = 1;
% [1 = save the workspace after solving for each cell voltage]

%% 5. CELL GEOMETRY

```

```

% Channel and membrane dimensions
Geometry.channel_length = 5.000e-2; % m [Length of channels]
Geometry.channel_width = 5.000e-3; % m [Width of channels]
Geometry.channel_height = 5.00e-4; % m [Depth of channels]
Geometry.mem_thick = 208e-6; % m [Membrane thickness]

% Discretization of the model domain
Grid_size.y_d_num = 15; % [Number of y-direction discretizations]
Grid_size.x_d_num = 50; % [Number of x-direction discretizations]

% For linear scaling
Geometry.x_d_min = 0.25/3;
% [Size of the smallest x-discretization with respect to the average.]
Geometry.y_d_min = 0.1;
% [Size of the smallest y-discretization with respect to the average.]

% For exponential scaling
Geometry.x_d_min_log = 1e-6; % m [Size of the smallest x-discretization.]
Geometry.y_d_min_log = 1e-5; % m [Size of the smallest y-discretization.]

%% 6. OPERATING PARAMETERS (DIRECLET BOUNDARY CONDITIONS)

% Anode side boundary conditions
BC.anode.elec_pot = 0; % V [Electric potential of anode]
BC.anode.x_vel_electrode = 0; % m/s [x-direction vel at electrodes]
BC.anode.x_vel_membrane = 0; % m/s [x-direction vel at membrane]
BC.anode.y_vel_electrode = 0; % m/s [y-direction vel at electrode]
BC.anode.y_vel_inlet = 0; % m/s [y-direction vel at inlet]
BC.anode.press_outlet = 0; % Pa [Pressure at inlet]

% Cathode side boundary conditions
BC.cathode.x_vel_electrode = 0; % m/s [x-direction vel at electrodes]
BC.cathode.x_vel_membrane = 0; % m/s [x-direction vel at membrane]
BC.cathode.y_vel_electrode = 0; % m/s [y-direction vel at electrode]
BC.cathode.y_vel_inlet = 0; % m/s [y-direction vel at inlet]
BC.cathode.press_outlet = 0; % Pa [Pressure at the inlet]

```

```

% List of cell voltages to evaluate.  If there is more than one, then a polarization curve is generated
% by solving teh model for each voltage sequentially.
Cathode_electric_potential = [ 1.6197 1.525 1.5 1.475 1.45 1.4 1.35 1.3 1.2 1.1 1.0 0.9 0.8 0.7 ...
    0.6 0.5 0.4 0.3 ]; % Exp54H

% Mean solution velocity at the inlet
anode_mean_inlet_velocity = 1.6666e-7 / (Geometry.channel_width*Geometry.channel_height);
% m/s [Mean inlet flow rate of fuel solution]
cathode_mean_inlet_velocity = 1.6666e-7 / (Geometry.channel_width*Geometry.channel_height);
% m/s [Mean inlet flow rate of oxidizer solution]

% The species mass fraction boundary conditions at the inlets are calculated from the fuel and oxidizer
% concentrations (set below). The inlet velocity boundary conditions are determined from the mean inlet
% velocities.

%% 7. MODEL PROPERTIES

% Note about the structure names in this section.  Typically one would make the top level catagories the
% largest and further subdivide for each subcatagory to minimize the number of fields in a structure.
% Here, the goal is to have structures in the form Model.discretization.property so that all of the
% properties of a particular discretization can be easily passed into a function... If they were
% organized as Model.property.discretization, this would be much more complicated.

% Species included in each phase.  Note that the membrane species list elements which appear on both the
% anode and cathode; if a species appears on both sides, its flux throught the membrane will be
% calculated.  Species should be in alphanumeric order.
Species.fuel.list = 'BH4'; 'BO2'; 'H2'; 'H2O'; 'Na'; 'OH' }; % 'BH3OH';
Species.oxidizer.list = 'H'; 'H2'; 'H2O'; 'H2O2'; 'Na'; 'O2'; 'SO4' };
Species.membrane.list = 'H2O'; 'Na' };
%intersect(Model.anode.species_list, Model.cathode.species_list);

% Store the total number of species in each list
Species.fuel.num = length(Species.fuel.list);
Species.oxidizer.num = length(Species.oxidizer.list);
Species.membrane.num = length(Species.membrane.list);

```

```
%% 8. SET UP SPECIES POINTERS
```

```
% Use the species lists to create pointers for the anode and cathode, which have different species. The  
% relative positions of the species are the same as in the lists.
```

```
% Create the anode species pointers
```

```
for s_i = 1 : Species.fuel.num  
    Pointer.anode.species.(char(Species.fuel.list(s_i))) = s_i;  
end
```

```
% Create the cathode species pointers
```

```
for s_i = 1 : Species.oxidizer.num  
    Pointer.cathode.species.(char(Species.oxidizer.list(s_i))) = s_i;  
end
```

```
% Create the membrane species pointers
```

```
for s_i = 1 : Species.membrane.num  
    Pointer.membrane.species.(char(Species.membrane.list(s_i))) = s_i;  
end
```

```
%% 9. FUEL PROPERTIES
```

```
% Include the mole densities (molarities) of each solute species. Do not include the solvent, which is  
% assumed to be water.
```

```
Fuel_mole_density.BH4    = 50E-3; % M [Molarity of NaBH4]  
Fuel_mole_density.BO2    = 1.0E-6; % M [Molarity of NaOH]  
Fuel_mole_density.OH     = 2.0E+0; % M [Molarity of NaBO2]  
Fuel_mole_density.H2     = 1.0E-6; % M [Molarity of H2]
```

```
% Ensure the solution is electrically neutral
```

```
Fuel_mole_density.Na = Fuel_mole_density.BH4 + Fuel_mole_density.BO2 + Fuel_mole_density.OH;
```

```
%% 10. OXIDIZER PROPERTIES
```

```
% Include the mole densities (molarities) of each ionic species. Do not include the solvent, which is
% assumed to be water.
```

```
Oxidizer_mole_density.H2O2 = 250E-3; % M [Molarity of H2O2]
Oxidizer_mole_density.Na   = 1.0E-6; % M [Molarity of Na+]
Oxidizer_mole_density.SO4  = 1.0E+0; % M [Molarity of SO4]
Oxidizer_mole_density.O2   = 2.67E-4; % M [Molarity of O2]
Oxidizer_mole_density.H2   = 1e-6; % M
```

```
% Ensure the solution is electrically neutral
```

```
Oxidizer_mole_density.H = 2 * Oxidizer_mole_density.SO4 - Oxidizer_mole_density.Na;
```

```
%% 11. INITIAL STATE
```

```
if Flags.setup.new_SV_initial == 1
```

```
% Concentrations in the channel are initially set to those of the inlet anode and cathode flows. To help
% the solver find a solution, the code in this section 1. sets the electric potential in the channels,
% imposing gradients 2. sets the mass fractions at the electrode and membrane interfaces 3. sets the
% initial guess for the inlet pressure, and imposes a gradient from inlet to outlet.
```

```
Initial.a_int_elec_pot_in   = 1.24;
Initial.a_int_elec_pot_out  = 1.24;
Initial.m_int_a_elec_pot_in = 1.24;
Initial.m_int_a_elec_pot_out = 1.24;
Initial.m_int_c_elec_pot_in = 1.25;
Initial.m_int_c_elec_pot_out = 1.25;
Initial.c_int_elec_pot_in   = 1.247;
Initial.c_int_elec_pot_out  = 1.247;
```

```
Initial.a_int_mass_fracs = [ (0.000067977539050 - 4e-5) (0.000000039211998 + 4e-5) ...
    0.000000001846454 0.926555707330665 0.042220459212564 0.031155814859269];
Initial.m_int_a_mass_fracs = [ (0.000067977539050 - 0e-5) (0.000000039211998 + 0e-5) ...
    0.000000001846454 0.926555707330665 0.042220459212564 0.031155814859269];
Initial.m_int_c_mass_fracs = [ 0.001876408651602 0.000000001876410 0.908382204023011 ...
    0.000316613634117 0.000000021399202 0.000007952577700 0.089416797837959];
```

```

Initial.c_int_mass_fracs = [ 0.001876408651602  0.000000001876410  0.908382204023011  0  ...
    0.0000000021399202  0.000007952577700  0.089416797837959];

% Set the initial guess for pressure at the inlet
Initial.inlet_press = 250; % Pa

% Set the initial guess for pressure at the membrane interfaces
Initial.m_int_a.press = BC.anode.press_outlet; % Pa
Initial.m_int_c.press = BC.cathode.press_outlet; % Pa

Initial.anode_press = 1e-3;
% Pa superimposed on the dominant gradient in the x-direction
Initial.cathode_press = -1e-3;
% Pa superimposed on the dominant gradient in the x-direction

Initial.m_int_a_y_velocity = 1E-6; % m/s
Initial.m_int_c_y_velocity = -1E-6; % m/s

end

%% 12. ERROR TOLERANCES

% These are used to check the initial solution vector and the final solution.
Tolerances.mole_fractions = 1E-5;
% unitless [Permitted deviation from sum(X_k) = 1]
Tolerances.mass_fractions = 1E-5;
% unitless [Permitted deviation from sum(Y_k) = 1]
Tolerances.non_neutrality = 1E-7;
% C/m^3 [Permitted deviation from charge density = 0]
Tolerances.mass_conservation = 5E-3;
% [Percent error for a control volume around the whole stack, ...
Tolerances.charge_conservation = 5E-4;

%% 14. CLEAR UNNECESSARY VARIABLES FROM THE WORKSPACE

clear Fuel.NaBH4_molarity Fuel.NaOH_molarity Fuel.NaBO2_molarity Oxidizer.H2O2_molarity ...

```

```

Oxidizer.HCl_molarity s_i smallest_x smallest_y

%% REFERENCES

% [1] DuPont Nafion materials specification sheet downloaded from DuPont website [2] Craig Urian
% experiments with PdIr coated plates

%% SUMMARY: DBFC_CONSTANTS_AND_PROPERTIES
% Purpose: Store universal constants and material properties which do not change.
% Author: Rick Stroman
% Date: 1 December 2011

%% DECLARE MODEL-WIDE GLOBAL VARIABLES

global Constants Properties Species

%% CONSTANTS

disp('Defining physical constants...')

Constants.e_charge      = 1.60218e-19; % C           [Charge on an electron. Ref 5, back cover]
Constants.e            = 2.71828;      % unitless    [Base of natural logarithm]
Constants.faraday      = 9.64853e7;    % C / kmol    [Coulombs of charge in 1 kmol of electrons. Ref 5]
Constants.avogadro     = 6.02214e26;   % number / kmol [Avogadro's number. Ref 5, back cover]
Constants.ideal_gas    = 8.31447e3;    % J / (K * kmol) [Ideal gas constant. Ref 5, back cover]
Constants.boltzman     = 1.38065e-23;  % J / K       [Boltzmann's constant. Ref 5, back cover]
Constants.pi           = 3.14159;     % unitless    [Ratio of circle circumference to diameter. Ref ?]
Constants.DH_A         = 1.2555;      %             [D-H constant value for water at 1 ATM and 300 K.]
Constants.DH_B         = 0.3965E10;   %             [D-H constant value for water at 1 ATM and 300 K.]
Constants.permittivity = 8.854187817620e-8; % F/m        [Permittivity of free space]
Constants.H2O_permeability = 80;
Constants.temperature  = 23+273.15;   % K

```

```
%% SPECIES PROPERTIES
```

```
disp('Defining species thermodynamic and electrochemical properties...')
```

```
% MOLAR MASS
```

```
Properties.molar_mass.Na      = 22.98976928; % kg/kmol [Ref 9]  
Properties.molar_mass.H       = 1.00794; % kg/kmol [Ref 9]  
Properties.molar_mass.OH      = 17.0073; % kg/kmol [Ref 9]  
Properties.molar_mass.BH4     = 14.843; % kg/kmol [Ref 9]  
Properties.molar_mass.Cl      = 35.453; % kg/kmol [Ref 9]  
Properties.molar_mass.BO2     = 42.810; % kg/kmol [Ref 9]  
Properties.molar_mass.H2O2    = 34.0147; % kg/kmol [Ref 9]  
Properties.molar_mass.H2O     = 18.0153; % kg/kmol [Ref 9]  
Properties.molar_mass.BH3OH   = 30.8424; % kg/kmol [sum of BH4 and O, Ref 9]  
Properties.molar_mass.H2      = 2.01588; % kg/kmol [Ref 9]  
Properties.molar_mass.O2      = 31.9988; % kg/kmol [Ref 9]  
Properties.molar_mass.SO4     = 96.063; % kg/kmol [Ref 9]
```

```
% ELECTRIC CHARGE
```

```
Properties.electric_charge.Na = +1;  
Properties.electric_charge.H   = +1;  
Properties.electric_charge.OH  = -1;  
Properties.electric_charge.BH4 = -1;  
Properties.electric_charge.Cl  = -1;  
Properties.electric_charge.BO2 = -1;  
Properties.electric_charge.H2O2 = 0;  
Properties.electric_charge.H2O = 0;  
Properties.electric_charge.e   = -1;  
Properties.electric_charge.BH3OH = -1;  
Properties.electric_charge.H2  = 0;  
Properties.electric_charge.O2  = 0;  
Properties.electric_charge.SO4 = -2;
```

```
% STANDARD APARENT MOLAR VOLUME AT INFINITE DILUTION
```

```
% Assuming T = 25 deg C and P = 1 bar.
```

```
Properties.apparent_volume.Na      = -1.11E-3; % m^3/kmol [Ref 1, Table 8.13, page 536]  
Properties.apparent_volume.H       = 0;      % m^3/kmol [Ref 1, Table 8.13, page 536... defined]  
Properties.apparent_volume.OH      = -4.18E-3; % m^3/kmol [Ref 1, Table 8.13, page 536]  
Properties.apparent_volume.BH4     = (-14.5E-3)/3; % m^3/kmol [GUESS --- VALUE NEEDED]  
Properties.apparent_volume.Cl      = 17.79E-3; % m^3/kmol [Ref 1, Table 8.13, page 536]  
Properties.apparent_volume.BO2     = -14.5E-3; % m^3/kmol [Ref 1, Table 8.13, page 536]  
Properties.apparent_volume.H2O2    = 22.17e-3; % m^3/kmol [Ref 10, page 2--]  
Properties.apparent_volume.H2O     = 0;      % m^3/kmol [Water is the solvent... 0 works out OK]  
Properties.apparent_volume.BH3OH   = 0;      % m^3/kmol [GUESS --- VALUE NEEDED]  
Properties.apparent_volume.H2      = 2.52E-2; % m^3/kmol [Ref 1, Table 9.14, page 653]  
Properties.apparent_volume.O2      = 3.038E-2; % m^3/kmol [Ref 1, Table 9.14, page 653]  
Properties.apparent_volume.SO4     = 24.8e-3; % m^3/kmol [Ref 16]
```

```
% SPECIES DIFFUSIVITIES IN WATER
```

```
% Diffusivities of each species in H2O, most at infinite dilution and 25 deg C.
```

```
Properties.diffusivity_in_water.Na  = 1.334E-9; % m^2/s  
%[Diffusivity of Na+ in H2O at infinite dilution and 25 deg C. Ref 6 Table 11.1]  
Properties.diffusivity_in_water.H   = 9.312E-9; % m^2/s  
%[Diffusivity of H+ in H2O at infinite dilution and 25 deg C. Ref 6 Table 11.1]  
Properties.diffusivity_in_water.OH  = 5.260E-9; % m^2/s  
%[Diffusivity of OH- in H2O at infinite dilution and 25 deg C. Ref 6 Table 11.1]  
Properties.diffusivity_in_water.BH4 = 1.5*2.42E-9; % m^2/s  
%[Diffusivity of BH4- in H2O and 2M NaOH at 25 deg C. Ref 4 page F19]  
Properties.diffusivity_in_water.Cl  = 2.032E-9; % m^2/s  
%[Diffusivity of Cl- in H2O at infinite dilution and 25 deg C. Ref 6 Table 11.1]  
Properties.diffusivity_in_water.BO2 = 8.14E-10; % m^2/s  
%[Diffusivity of BO2- in H2O at infinite dilution and 25 deg C. Ref 14]  
Properties.diffusivity_in_water.H2O2 = 1.61E-9; % m^2/s  
%[Diffusivity of H2O2 in H2O from ref 18... at 25 deg C ]  
Properties.diffusivity_in_water.H2O = 0 ; % m^2/s  
%[H2O diffusion balances solute diffusion... this value is irrelevant, but keeps things from blowing up]  
Properties.diffusivity_in_water.BH3OH = 7.48E-10; % m^2/s
```

```

%[Diffusivity in 0.02 M DMAB and 1 M NaOH at 20 deg C. Ref 12]
Properties.diffusivity_in_water.H2      = 4.5E-9;    % m^2/s [Diffusivity of H2 in H2O Ref 17]
Properties.diffusivity_in_water.O2      = 1.97E-9;    % m^2/s [Diffusivity of O2 in H2O Ref 17]
Properties.diffusivity_in_water.SO4     = 6.25e-10;   % m^2/s [Diffusivity of SO4-2 in water . Ref 15]

%% MEMBRANE PROPERTIES

Properties.membrane.electro_drag        = 9.2;        % unitless
%[Mole H2O transported per mole Na+, for fully hydrated (lambda = 18) membrane in Na+ form. Ref 8]
Properties.membrane.Na_mobility         = 2.7E-8;     % m^2 / (V s)
%[Mobility of Na+ in a fully hydrated Nafion 115 membrane where X_Na = 1.0. Ref 8]
Properties.membrane.H_mobility          = 1.49E-7;    % m^2 / (V s)
%[Mobility of H+ in a fully hydrated Nafion 115 membrane where X_H =1.0. Ref 8]
Properties.membrane.SO3_density         = 1.13;       % kmol/m^3
%[Density of sulfonic acid groups in fully hydrated Nafion 115. Ref 8]
Properties.membrane.k                   = 0.20;       % unitless
%[Na+ and H+ mobility interaction factor in fully hydrated Nafion 115, Ref 8]
Properties.membrane.hydration           = 18.4;       % H2O per SO3-
%[Number of water molecules per sulfonic acid group in the membrane at full hydration in the Na+ form in
%Nafion 115, Ref 8]
Properties.membrane.H2O_diffusivity     = 3.5E-10;    % m^2/s
Properties.membrane.permeability        = 1.7e-14;    % m Pa^-1 s^-1
%[Filtration coefficient for water through Nafion 125 at 20 deg C in 3 M NaCl, Ref 13]

%% SOLVENT PROPERTIES

% Need to do this twice becuae of the way Properties is passed into functions...
Properties.anode.density_water          = FUNC_WATER_DENSITY(Constants.temperature);
Properties.cathode.density_water       = FUNC_WATER_DENSITY(Constants.temperature);

%% STORE SPECIES PROPERTIES IN VECTORS TO SIMPLIFY USE OF PROPERTIES IN LOOPS OVER ALL SPECIES

for species_position = 1 : Species.fuel.num
    species_name = char(Species.fuel.list(species_position));
    Properties.anode.Molar_mass(species_position,1) = Properties.molar_mass.(species_name);
    Properties.anode.Electric_charge(species_position,1) = Properties.electric_charge.(species_name);

```

```

Properties.anode.Ionic_diameter(species_position,1) = Properties.ionic_diameter.(species_name);
Properties.anode.Apparent_volume(species_position,1) = Properties.apparent_volume.(species_name);
Properties.anode.Diffusivities(species_position,1) = Properties.diffusivity_in_water.(species_name);
end

for species_position = 1 : Species.oxidizer.num
    species_name = char(Species.oxidizer.list(species_position));
    Properties.cathode.Molar_mass(species_position,1) = Properties.molar_mass.(species_name);
    Properties.cathode.Electric_charge(species_position,1) = Properties.electric_charge.(species_name);
    Properties.cathode.Ionic_diameter(species_position,1) = Properties.ionic_diameter.(species_name);
    Properties.cathode.Apparent_volume(species_position,1) = Properties.apparent_volume.(species_name);
    Properties.cathode.Diffusivities(species_position,1) = ...
        Properties.diffusivity_in_water.(species_name);
end

for species_position = 1 : Species.membrane.num
    species_name = char(Species.membrane.list(species_position));
    Properties.membrane.Molar_mass(species_position,1) = Properties.molar_mass.(species_name);
    Properties.membrane.Electric_charge(species_position,1) = Properties.electric_charge.(species_name);
end

%% CALCULATE 1 / SEVERAL PROPERTIES HERE INSTEAD OF IN THE MAIN FUNCTION BECAUSE DIVISION IS SLOW

Properties.anode.one_over_molar_mass = 1 ./ Properties.anode.Molar_mass;
Properties.cathode.one_over_molar_mass = 1 ./ Properties.cathode.Molar_mass;
Properties.anode.Appar_vol_over_Molar_mass = (Properties.anode.Apparent_volume ...
    ./ Properties.anode.Molar_mass)';
Properties.cathode.Appar_vol_over_Molar_mass = (Properties.cathode.Apparent_volume ...
    ./ Properties.cathode.Molar_mass)';

%% CALCULATE TRANSPOSES OF SEVERAL PROPERTIES HERE BECAUSE DIVISION IS SLOW

Properties.anode.Molar_mass_T = Properties.anode.Molar_mass';
Properties.cathode.Molar_mass_T = Properties.cathode.Molar_mass';

Properties.anode.Electric_charge_T = Properties.anode.Electric_charge';

```

```

Properties.cathode.Electric_charge_T = Properties.cathode.Electric_charge';

Constants.FoRT = Constants.faraday / (Constants.ideal_gas * Constants.temperature);
Constants.RToF = 1/Constants.FoRT;

%% CLEAR UNNECESSARY VARIABLES FROM THE WORKSPACE

clear species species_position

%% REFERENCES

% [1] Ottonello, G., Principles of Geochemistry. 1997, New York: Columbia University Press. xii, 894 p.
% [2] Sodaye, S., C. Agarwal, et al. (2008). "Study on multicomponent diffusion of ions in poly
% (perfluorosulfonated) ion-exchange membrane using radiotracers." Journal of Membrane Science 314(1-2):
% 221-225.
% [3] Bird, Steward and Lightfoot
% [4] Santos, D. M. F. and C. A. C. Sequeira (2010). "Chronopotentiometric Investigation of Borohydride
% Oxidation at a Gold Electrode." Journal of the Electrochemical Society 157(1): F16-F21.
% [5] Bard, A. J. and L. R. Faulkner (2001). Electrochemical methods : fundamentals and applications. New
% York, Wiley.
% [6] Newman, J. S. and K. E. Thomas-Alyea (2004). Electrochemical systems. Hoboken, N.J., J. Wiley.
% [7] DuPont Nafion 115 specifications sheet. www2.dupont.com/FuelCells/en_US/assets/downloads/dfc101.pdf
% [8] Okada, T., S. Moller-Holst, et al. (1998). "Transport and equilibrium properties of Nafion (R)
% membranes with H+ and Na+ ions." Journal of Electroanalytical Chemistry 442(1-2): 137-145.
% [9] NIST Web Book. http://webbook.nist.gov
% [10] Schumb, W. C. (1955). Hydrogen peroxide. New York,, Reinhold Pub. Corp.
% [11] Macpherson, J. V. and P. R. Unwin (1997). "Determination of the diffusion coefficient of hydrogen
% in aqueous solution using single and double potential step chronoamperometry at a disk
% ultramicroelectrode." Analytical Chemistry 69(11): 2063-2069.
% [12] Nagle, L. C. and J. F. Rohan (2005). "Investigation of DMAB oxidation at a gold microelectrode in
% base." Electrochemical and Solid State Letters 8(5): C77-C80.
% [13] Evans, C. E., R. D. Noble, et al. (2006). "Role of conditioning on water uptake and hydraulic
% permeability of Nafion (R) membranes." Journal of Membrane Science 279(1-2): 521-528.
% [14] Cloutier, C. R., A. Alfantazi, et al. (2007). "Physicochemical Transport Properties of Aqueous
% Sodium Metaborate Solutions for Sodium Borohydride Hydrogen Generation and Storage and Fuel Cell
% Applications." Thermec 2006 Supplement 15-17: 267-274.

```

```

% [15] Nielsen, J.M., A.W. Adamson, and J.W. Cobble, The Self-Diffusion Coefficients of the Ions in
% Aqueous Sodium Chloride and Sodium Sulfate at 25-Degrees. Journal of the American Chemical Society,
% 1952. 74(2): p. 446-451.
% [16] Poisson, A. and J. Chanu, Semi-Empirical Equations for the Partial Molar Volumes of Some Ions in
% Water and Seawater. Mar. Chem., 1980. 8: p. 289-298.
% [17] C.R. Wilke, P. Chang, Aiche J, 1 (1955) 264-270. [18] S.B. Hall, E.A. Khudaish, A.L.
% Hart, Electrochim Acta, 43 (1998) 579-588.

```

```

%% SUMMARY: DBFC_SETUP_REACTION_RATES

```

```

% Purpose: Read user provided stoichiometric and rate parameters for the Arxn taking place at the RDE.
% Collect those stoichiometric and rate parameters into the structures expected by the reaction rate
% function in the model. Author: Rick Stroman

```

```

%% NOTES

```

```

% 1. ALL species must be given a stoichiometry and concentration dependence (both forward and reverse).
% If a species doesn't participate, set its stoichiometry to zero. If a species which doesn't exist in
% the model is given a value or a species which exists isn't given a value, the model will throw an
% error. Assign a stoichiometric coefficient to the electrons, though they are excluded when the
% "Reaction_stoich" vector is built so it does not cause problems calculating the mass fluxes.

```

```

%% 1. SETUP GLOBAL VARIABLES

```

```

global Arxn Crxn Constants Pointer Species

```

```

%% 2. ANODE REACTION RATES

```

```

% Overall anode parameters

```

```

Arxn.param.area_ratio = 2.73; % unitless [Ratio electrochemical/geometric surface area]

```

```

% -----%
% Reaction #1... 1 BH4- + 8 OH- <--> 1 BO2- + 6 H2O + 8 e-

```

```

% -----%

Arxn.rxn(1).active = 1;

% Species stoichiometries
Arxn.rxn(1).stoich.BH4 = -1;
Arxn.rxn(1).stoich.BO2 = 1;
Arxn.rxn(1).stoich.e = 8;
Arxn.rxn(1).stoich.H2 = 0;
Arxn.rxn(1).stoich.H2O = 6;
Arxn.rxn(1).stoich.Na = 0;
Arxn.rxn(1).stoich.OH = -8;

% Rate constants, electron transfer coefficients and standard half-cell potential.
Arxn.rxn(1).k_f = 1.25*0.00704; % m^4 / (kmol s) [Anodic direction reaction rate constant]
Arxn.rxn(1).beta_f = 0.098; % unitless [Anodic charge transfer coefficient]
Arxn.rxn(1).phi_0 = -1.240; % V [Standard half-cell potential]
Arxn.rxn(1).e_rds = 1; % unitless [Number of electrons transferred in rate determining step]

Arxn.rxn(1).beta_r = 1 - Arxn.rxn(1).beta_f; % unitless [Cathodic charge transfer coefficient]
Arxn.rxn(1).k_r = Arxn.rxn(1).k_f * exp( Arxn.rxn(1).e_rds * Constants.faraday / ...
    (Constants.ideal_gas * Constants.temperature) * Arxn.rxn(1).phi_0 );

% Concentration dependencies (assumed to be first order) for the cathodic and anodic directions
Arxn.rxn(1).conc_dependence_f = [ Pointer.anode.species.BH4 ];
Arxn.rxn(1).conc_dependence_r = [ Pointer.anode.species.BO2 ];

% -----%
% Reaction #2... 1 H2 + 2 OH- <--> 2 H2O + 2 e-
% -----%

Arxn.rxn(2).active = 0;

% Species stoichiometries
Arxn.rxn(2).stoich.BH4 = 0;
Arxn.rxn(2).stoich.BO2 = 0;

```

```

Arxn.rxn(2).stoich.e = 2;
Arxn.rxn(2).stoich.H2 = -1;
Arxn.rxn(2).stoich.H2O = 2;
Arxn.rxn(2).stoich.Na = 0;
Arxn.rxn(2).stoich.OH = -2;

% Rate constants, electron transfer coefficients and standard half-cell potential.
Arxn.rxn(2).k_r = 1.0e-15; % [Cathodic direction reaction rate constant]
Arxn.rxn(2).beta_r = 0.5000; % unitless [Cathodic charge transfer coefficient]
Arxn.rxn(2).phi_0 = -0.828; % V [Standard half-cell potential]
Arxn.rxn(2).e_rds = 1; % unitless [Number of electrons transferred in the rate determining step]

Arxn.rxn(2).beta_f = 1 - Arxn.rxn(2).beta_r; % unitless [Cathodic charge transfer coefficient]
Arxn.rxn(2).k_f = Arxn.rxn(2).k_r / exp( Arxn.rxn(2).e_rds * Constants.faraday / ...
    (Constants.ideal_gas * Constants.temperature) * Arxn.rxn(2).phi_0 );

% Concentration dependencies (assumed to be first order) for the forward and reverse rates
Arxn.rxn(2).conc_dependence_f = [ Pointer.anode.species.H2 Pointer.anode.species.OH ];
Arxn.rxn(2).conc_dependence_r = [ ];

% -----%
% Reaction #3... BH4- + 2 H2O --> BO2- + 4 H2
% -----%

Arxn.rxn(3).active = 0;

% Species stoichiometries
Arxn.rxn(3).stoich.BH4 = -1;
Arxn.rxn(3).stoich.BO2 = 1;
Arxn.rxn(3).stoich.e = 0;
Arxn.rxn(3).stoich.H2 = 4;
Arxn.rxn(3).stoich.H2O = -2;
Arxn.rxn(3).stoich.Na = 0;
Arxn.rxn(3).stoich.OH = 0;

% Rate constants, electron transfer coefficients and standard half-cell potential.

```

```

Arxn.rxn(3).k_f      = 1.5*2.06e-4; % [Anodic direction reaction rate constant]
Arxn.rxn(3).beta_f  = 0; % unitless [Anodic charge transfer coefficient]
Arxn.rxn(3).phi_0   = 0; % V        [Standard half-cell potential]
Arxn.rxn(3).e_rds   = 0; % unitless [Number of e- transferred in the rate determining step]

Arxn.rxn(3).beta_r  = 0;
Arxn.rxn(3).k_r     = 0;

% Concentration dependencies (assumed to be first order) for the forward and reverse rates
Arxn.rxn(3).conc_dependence_f = [ Pointer.anode.species.BH4 ];
Arxn.rxn(3).conc_dependence_r = [ ];

%% 3. CATHODE REACTION RATES

% Overall anode parameters

Crxn.param.area_ratio = 4.11; % unitless [Ratio electrochemical/geometric surface area] 1.0

% -----%
% Reaction #1...   2 H2O <-->  1 H2O2 + 2 H+ + 2e-
% -----%

Crxn.rxn(1).active = 1;

% Species stoichiometries
Crxn.rxn(1).stoich.e      = 2;
Crxn.rxn(1).stoich.H      = 2;
Crxn.rxn(1).stoich.H2O    = -2;
Crxn.rxn(1).stoich.H2O2   = 1;
Crxn.rxn(1).stoich.Na     = 0;
Crxn.rxn(1).stoich.O2     = 0;
Crxn.rxn(1).stoich.SO4    = 0;
Crxn.rxn(1).stoich.H2     = 0;

% Rate constants, electron transfer coefficients and standard half-cell potential.
Crxn.rxn(1).k_r          = 7.54e-3; % m^4 / (kmol s) [Cathodic direction reaction rate constant]

```

```

Crxn.rxn(1).beta_r = 0.455;      % unitless [Cathodic charge transfer coefficient]
Crxn.rxn(1).phi_0 = 1.763;      % V      [Standard half-cell potential]
Crxn.rxn(1).e_rds = 1;          % unitless [Number of electrons transferred in the rate dermining step]

Crxn.rxn(1).beta_f = 1 - Crxn.rxn(1).beta_r; % unitless [Cathodic charge transfer coefficient]
Crxn.rxn(1).k_f = Crxn.rxn(1).k_r / ( exp( Crxn.rxn(1).e_rds * Constants.faraday / ...
                                     (Constants.ideal_gas * Constants.temperature) * Crxn.rxn(1).phi_0 ) );

% Concentration dependencies (assumed to be first order) for the forward and reverse rates
Crxn.rxn(1).conc_dependence_r = [ Pointer.cathode.species.H2O2 Pointer.cathode.species.H ...
                                Pointer.cathode.species.H];
Crxn.rxn(1).conc_dependence_f = [ ];

% -----
% Reaction #2...  1 H2O2 <--> 1 O2 + 2 H+ + 2e-
% -----

Crxn.rxn(2).active = 0;

% Species stoichiometries
Crxn.rxn(2).stoich.e = 2;
Crxn.rxn(2).stoich.H = 2;
Crxn.rxn(2).stoich.H2O = 0;
Crxn.rxn(2).stoich.H2O2 = -1;
Crxn.rxn(2).stoich.Na = 0;
Crxn.rxn(2).stoich.O2 = 1;
Crxn.rxn(2).stoich.SO4 = 0;
Crxn.rxn(2).stoich.H2 = 0;

% Rate constants, electron transfer coefficients and standard half-cell potential.
Crxn.rxn(2).k_f = 1e-6; % m^4 / (kmol s) (Abruna on Au, 5 mM BH4 in NaOH)%2.7397e-2;
Crxn.rxn(2).beta_f = 0.5; % unitless [Anodic charge transfer coefficient]
Crxn.rxn(2).phi_0 = 0.695; % V      [Standard half-cell potential]
Crxn.rxn(2).e_rds = 1; % unitless [Number of electrons transferred in rate dermining step]

Crxn.rxn(2).beta_r = 1 - Crxn.rxn(2).beta_f; % unitless [Cathodic charge transfer coefficient]

```

```

Crxn.rxn(2).k_r      = Crxn.rxn(2).k_f * exp( Crxn.rxn(2).e_rds * Constants.faraday / ...
                    (Constants.ideal_gas * Constants.temperature) * Crxn.rxn(2).phi_0 );

% Concentration dependencies (assumed to be first order) for the forward and reverse rates
Crxn.rxn(2).conc_dependence_r = [ Pointer.cathode.species.O2; Pointer.cathode.species.H ];
Crxn.rxn(2).conc_dependence_f = [ Pointer.cathode.species.H2O2 ];

% -----
% Reaction #3...  2 H2O2 --> 2 H2O + O2
% -----

Crxn.rxn(3).active = 0;

% Species stoichiometries
Crxn.rxn(3).stoich.e      = 0;
Crxn.rxn(3).stoich.H      = 0;
Crxn.rxn(3).stoich.H2O    = 2;
Crxn.rxn(3).stoich.H2O2  = -2;
Crxn.rxn(3).stoich.Na     = 0;
Crxn.rxn(3).stoich.O2     = 1;
Crxn.rxn(3).stoich.SO4    = 0;
Crxn.rxn(3).stoich.H2     = 0;

% Rate constants, electron transfer coefficients and standard half-cell potential.
Crxn.rxn(3).k_r          = 0;    % [Cathodic direction reaction rate constant] Pt: 8e-3
Crxn.rxn(3).beta_r      = 0;    % unitless [Cathodic charge transfer coefficient] 0.50 Pt: 0.45
Crxn.rxn(3).phi_0       = 0;    % V [Standard half-cell potential]
Crxn.rxn(3).e_rds       = 0;    % unitless [Number of electrons transferred in the rate determining step]

Crxn.rxn(3).beta_f      = 0;    % unitless [Cathodic charge transfer coefficient]
Crxn.rxn(3).k_f         = 6.34E-4; % [Forward direction reaction rate constant]

% Concentration dependencies (assumed to be first order) for the forward and reverse rates
Crxn.rxn(3).conc_dependence_r = [ Pointer.cathode.species.O2 ];
Crxn.rxn(3).conc_dependence_f = [ Pointer.cathode.species.H2O2 ];

```

```

% ----- %
% Reaction #4... 1 H2 <--> 2 H+ + 2 e-
% ----- %

Crxn.rxn(4).active = 1;

% Species stoichiometries
Crxn.rxn(4).stoich.e      = 2;
Crxn.rxn(4).stoich.H      = 2;
Crxn.rxn(4).stoich.H2O    = 0;
Crxn.rxn(4).stoich.H2O2   = 0;
Crxn.rxn(4).stoich.Na     = 0;
Crxn.rxn(4).stoich.O2     = 0;
Crxn.rxn(4).stoich.SO4    = 0;
Crxn.rxn(4).stoich.H2     = -1;

% Rate constants, electron transfer coefficients and standard half-cell potential.
Crxn.rxn(4).k_r          = 0.5*2.37e-09; % [Cathodic direction reaction rate constant]
Crxn.rxn(4).beta_r      = .8*0.1764;    % unitless [Cathodic charge transfer coefficient]
Crxn.rxn(4).phi_0       = 0.000;        % V [Standard half-cell potential]
Crxn.rxn(4).e_rds       = 2;            % unitless [Number of electrons transferred in rate determining step]

Crxn.rxn(4).beta_f      = 1 - Crxn.rxn(4).beta_r; % unitless [Cathodic charge transfer coefficient]
Crxn.rxn(4).k_f         = Crxn.rxn(4).k_r / exp( Crxn.rxn(4).e_rds * Constants.faraday / ...
                                                (Constants.ideal_gas * Constants.temperature) * Crxn.rxn(4).phi_0 );

% Concentration dependencies (assumed to be first order) for the forward and reverse rates
Crxn.rxn(4).conc_dependence_r = [ Pointer.cathode.species.H Pointer.cathode.species.H];
Crxn.rxn(4).conc_dependence_f = [ Pointer.cathode.species.H2 ];

%% 4. PROCESS THE REACTION RATE PARAMETERS

% Store the stoichiometric coefficients in a vector with species organized in the same order as in the
% species list.

for r_i = 1:length(Arxn.rxn)

```

```

Arxn.rxn(r_i).Reaction_stoich = zeros(1,Species.fuel.num);

for s_i = 1:Species.fuel.num
    Arxn.rxn(r_i).Reaction_stoich(1,s_i) = Arxn.rxn(r_i).stoich.(char(Species.fuel.list(s_i)));
end

end

for r_i = 1:length(Crxn.rxn)

    Crxn.rxn(r_i).Reaction_stoich = zeros(1,Species.oxidizer.num);

    for s_i = 1:Species.oxidizer.num
        Crxn.rxn(r_i).Reaction_stoich(1,s_i) = Crxn.rxn(r_i).stoich.(char(Species.oxidizer.list(s_i)));
    end

end

%% SUMMARY: DBFC_INITIALIZE
% Purpose: Create the solution vector with the initial values at which the solver will start.
% Author: Rick Stroman

%% 1. DECLARE MODEL-WIDE GLOBAL VARIABLES AND DISPLAY STATUS MESSAGE

global Model Pointer Names Scales Units BC Initial SV_fail SV_fail_lg Geometry Flags Solver ...
        Jacobian Grid_size Species

%% 1. PRINT SOME KEY PARTS OF THE SETUP

disp(' ')
disp('SELECTED MODEL INPUT PARAMETERS')
disp('-----')

```

```

disp(['Number of x-discretizations: ' num2str(Grid_size.x_d_num)])
disp(['Number of y-discretizations: ' num2str(Grid_size.y_d_num)])
disp(' ');
disp(['Channel height: ', num2str(Geometry.channel_height), ' m'])
disp(['Channel width: ', num2str(Geometry.channel_width), ' m'])
disp(['Channel length: ', num2str(Geometry.channel_length), ' m'])
disp(['Membrane thickness: ', num2str(Geometry.mem_thick), ' m'])
disp(' ');
disp(['Model Temperature: ', num2str(Constants.temperature), ' K'])
disp(' ');
disp('Anode concentrations:')
disp(['[BH4-]: ', num2str(Fuel_mole_density.BH4), ' M'])
disp(['[BO2-]: ', num2str(Fuel_mole_density.BO2), ' M'])
disp(['[OH-]: ', num2str(Fuel_mole_density.OH), ' M'])
disp(['[Na+]: ', num2str(Fuel_mole_density.Na), ' M'])
disp(' ');
disp('Cathode concentrations:')
disp(['[H2O2]: ', num2str(Oxidizer_mole_density.H2O2), ' M'])
disp(['[SO4-]: ', num2str(Oxidizer_mole_density.SO4), ' M'])
disp(['[H+]: ', num2str(Oxidizer_mole_density.H), ' M'])
disp(['[Na+]: ', num2str(Oxidizer_mole_density.Na), ' M'])
disp(' ');
disp(['Fuel inlet flow rate: ', num2str(anode_flowrate_inlet), ' m^3 / s'])
disp(['Oxidizer inlet flow rate: ', num2str(cathode_flowrate_inlet), ' m^3 / s'])
disp(' ');
disp(['Fuel inlet velocity: ', num2str(anode_mean_inlet_velocity), ' m/s'])
disp(['Oxidizer inlet velocity: ', num2str(cathode_mean_inlet_velocity), ' m/s'])
disp(' ');
fprintf('%-32s %-3e\n', 'Anode anodic rate constant: ', Arxn.rxn(1).k_f)
fprintf('%-32s %-3e\n', 'Cathode cathodic rate constant: ', Crxn.rxn(1).k_r)
disp(' ');
disp(['Cell potential(s): ', num2str(Cathode_electric_potential), ' V'])
disp(' ');
if Flags.setup.new_SV_initial == 1;
    disp('Starting simulation with the standard guess')
else

```

```

    disp(['Starting simulation with solution file (' Solver.SV_initial_filename ')'])
end
disp('-----')
disp(' ')

disp(' ')
disp('Initializing the model...')

%% 2. ESTABLISH POINTERS, NAMES, SCALES CONSTRAINTS AND UNITS

disp('Creating pointers, state variable names, units, and scales...')

% Initialize the counter that tracks which element in the solution vector is being operated on.
SV_position = 1;

% Initialize the variables containing the number of state variables in each part of the cell.
Grid_size.anode.state_vars_num = 0;
Grid_size.cathode.state_vars_num = 0;
Grid_size.membrane.state_vars_num = 0;

for x_d = 1 : Grid_size.x_d_num

    x_d_string = strcat('(', num2str(x_d), ')');

    % -----%
    % y-Discretization "a_int". Anode electrolyte solution
    % -----%

    % Electric Potential
    Pointer.a_int.elec_pot(x_d) = SV_position;
    Names{Pointer.a_int.elec_pot(x_d),1} = strcat('A_int.elec_pot', x_d_string);
    Scales.SV(Pointer.a_int.elec_pot(x_d),1) = Solver.SV_scale.elec_pot;
    Scales.dSV(Pointer.a_int.elec_pot(x_d),1) = Solver.dSV_scale.electroneutrality;
    Units{Pointer.a_int.elec_pot(x_d),1} = 'V';
    Solver.kinsol.Constraints(Pointer.a_int.elec_pot(x_d),1) = 0; % 0 --> none, 1 --> >= 0, 2 --> >0
    Grid_size.anode.state_vars_num = Grid_size.anode.state_vars_num + 1;

```

```

SV_position                                = SV_position + 1;

% Mass Fractions
Pointer.a_int.mass_fracs(x_d) = SV_position;
for species = 1 : Species.fuel.num
    Names{Pointer.a_int.mass_fracs(x_d) + species - 1,1} = strcat('A_int.mass_fr.' , ...
        char(Species.fuel.list(species)), x_d_string );
end
Pointer_range.a_int = Pointer.a_int.mass_fracs(x_d) :
    Pointer.a_int.mass_fracs(x_d) + Species.fuel.num - 1;
Scales.SV(Pointer_range.a_int,1)          = Solver.SV_scale.mass_frac_anode; % Scale for mass fracs
Scales.dSV(Pointer_range.a_int,1)         = Solver.dSV_scale.species_flux_balance_a;
Units(Pointer_range.a_int,1)              = {'none'};
Solver.kinsol.Constraints(Pointer_range.a_int,1) = 1; % 0 --> none, 1 --> >= 0, 2 --> >0
Grid_size.anode.state_vars_num            = Grid_size.anode.state_vars_num + Species.fuel.num;
SV_position                                = SV_position + Species.fuel.num;

% -----%
% y-Discretization "asln". Anode electrolyte solution
% -----%

for y_d = 1 : Grid_size.y_d_num

    x_y = strcat('(' , num2str(x_d) , ',' , num2str(y_d) , ')');

    % Electric Potential
    Pointer.asln.elec_pot(x_d,y_d)          = SV_position;
    Names{Pointer.asln.elec_pot(x_d,y_d),1} = strcat('Asln.elec_pot',x_y);
    Scales.SV(Pointer.asln.elec_pot(x_d,y_d),1) = Solver.SV_scale.elec_pot;
    Scales.dSV(Pointer.asln.elec_pot(x_d,y_d),1) = Solver.dSV_scale.electroneutrality;
    Units{Pointer.asln.elec_pot(x_d,y_d),1} = 'V';
    Solver.kinsol.Constraints(Pointer.asln.elec_pot(x_d,y_d),1) = 0; % 0 --> none, 1 --> >= 0
    Grid_size.anode.state_vars_num          = Grid_size.anode.state_vars_num + 1;
    SV_position                              = SV_position + 1;

    % Mass Fractions

```

```

Pointer.asln.mass_fracs(x_d,y_d) = SV_position;
for species = 1 : Species.fuel.num
    Names{Pointer.asln.mass_fracs(x_d,y_d) + species - 1,1} = ...
        strcat('Asln.mass_fr.', char(Species.fuel.list(species)), x_y );
end
Pointer_range.asln = Pointer.asln.mass_fracs(x_d,y_d) : ...
    Pointer.asln.mass_fracs(x_d,y_d) + Species.fuel.num - 1;
Scales.SV(Pointer_range.asln,1) = Solver.SV_scale.mass_frac_anode; % Scale for mass fractions
Scales.dSV(Pointer_range.asln,1) = Solver.dSV_scale.species_cons_a;
Units(Pointer_range.asln,1) = {'none'}; % Mass fraction is unitless
Solver.kinsol.Constraints(Pointer_range.asln,1) = 0; % 0 --> none, 1 --> >= 0
Grid_size.anode.state_vars_num = Grid_size.anode.state_vars_num + Species.fuel.num;
SV_position = SV_position + Species.fuel.num;

% x-Velocity
Pointer.asln.x_vel(x_d,y_d) = SV_position;
Names{Pointer.asln.x_vel(x_d,y_d),1} = strcat('Asln.x_velocity',x_y);
Scales.SV(Pointer.asln.x_vel(x_d,y_d),1) = Solver.SV_scale.x_vel;
Scales.dSV(Pointer.asln.x_vel(x_d,y_d),1) = Solver.dSV_scale.x_momentum_cons;
Units{Pointer.asln.x_vel(x_d,y_d),1} = 'm/s';
Solver.kinsol.Constraints(Pointer.asln.x_vel(x_d,y_d),1) = 0; % None
Grid_size.anode.state_vars_num = Grid_size.anode.state_vars_num + 1;
SV_position = SV_position + 1;

% y-Velocity
Pointer.asln.y_vel(x_d,y_d) = SV_position;
Names{Pointer.asln.y_vel(x_d,y_d),1} = strcat('Asln.y_velocity',x_y);
Scales.SV(Pointer.asln.y_vel(x_d,y_d),1) = Solver.SV_scale.y_vel;
Scales.dSV(Pointer.asln.y_vel(x_d,y_d),1) = Solver.dSV_scale.y_momentum_cons;
Units{Pointer.asln.y_vel(x_d,y_d),1} = 'm/s';
Solver.kinsol.Constraints(Pointer.asln.y_vel(x_d,y_d),1) = 0; % None
Mass_matrix_diag(Pointer.asln.y_vel(x_d,y_d)) = 1; % Differential (residual is d v_y / dt)
Grid_size.anode.state_vars_num = Grid_size.anode.state_vars_num + 1;
SV_position = SV_position + 1;

% Pressure

```

```

Pointer.asln.press(x_d,y_d) = SV_position;
Names{Pointer.asln.press(x_d,y_d),1} = strcat('Asln.pressure',x_y);
Scales.SV(Pointer.asln.press(x_d,y_d),1) = Solver.SV_scale.press;
Scales.dSV(Pointer.asln.press(x_d,y_d),1) = Solver.dSV_scale.mass_cons;
Units{Pointer.asln.press(x_d,y_d),1} = 'Pa';
Solver.kinsol.Constraints(Pointer.asln.press(x_d,y_d),1) = 0; % None
Grid_size.anode.state_vars_num = Grid_size.anode.state_vars_num + 1;
SV_position = SV_position + 1;

end

% -----%
% y-Discretization "m_int_a". Anode electrolyte solution/membrane interface
% -----%

% Electric Potential
Pointer.m_int_a.elec_pot(x_d) = SV_position;
Names{Pointer.m_int_a.elec_pot(x_d),1} = strcat('M_int_a.elec_pot', x_d_string);
Scales.SV(Pointer.m_int_a.elec_pot(x_d),1) = Solver.SV_scale.elec_pot;
Scales.dSV(Pointer.m_int_a.elec_pot(x_d),1) = Solver.dSV_scale.electroneutrality;
Units{Pointer.m_int_a.elec_pot(x_d),1} = 'V';
Solver.kinsol.Constraints(Pointer.m_int_a.elec_pot(x_d),1) = 0; % None
Grid_size.anode.state_vars_num = Grid_size.anode.state_vars_num + 1;
SV_position = SV_position + 1;

% Mass Fractions
Pointer.m_int_a.mass_fracs(x_d) = SV_position;
for species = 1 : Species.fuel.num
    Names{Pointer.m_int_a.mass_fracs(x_d) + species - 1,1} = ...
        strcat('M_int_a.mass_fr.', char(Species.fuel.list(species)), x_d_string);
end
Pointer_range.m_int_a = Pointer.m_int_a.mass_fracs(x_d) : ...
    Pointer.m_int_a.mass_fracs(x_d) + Species.fuel.num - 1;
Scales.SV(Pointer_range.m_int_a,1) = Solver.SV_scale.mass_frac_anode; % Scale for mass fractions
Scales.dSV(Pointer_range.m_int_a,1) = Solver.dSV_scale.species_flux_balance_a;
Units(Pointer_range.m_int_a,1) = {'none'}; % Mass fraction is unitless

```

```

Solver.kinsol.Constraints(Pointer_range.m_int_a,1) = 0; % 0 --> none, 1 --> >= 0
Grid_size.anode.state_vars_num = Grid_size.anode.state_vars_num + Species.fuel.num;
SV_position = SV_position + Species.fuel.num;

% Pressure
Pointer.m_int_a.press(x_d) = SV_position;
Names{Pointer.m_int_a.press(x_d),1} = strcat('M_int_a.pressure', x_d_string);
Scales.SV(Pointer.m_int_a.press(x_d),1) = Solver.SV_scale.press;
Scales.dSV(Pointer.m_int_a.press(x_d),1) = Solver.dSV_scale.mass_flux_balance;
Units{Pointer.m_int_a.press(x_d),1} = 'Pa';
Solver.kinsol.Constraints(Pointer.m_int_a.press(x_d),1) = 0; % 0 --> none, 1 --> >= 0, 2 --> >0
Grid_size.anode.state_vars_num = Grid_size.anode.state_vars_num + 1;
SV_position = SV_position + 1;

% -----%
% y-Discretization "m_int_c". Cathode electrolyte solution/membrane interface
% -----%

% Electric Potential
Pointer.m_int_c.elec_pot(x_d) = SV_position;
Names{Pointer.m_int_c.elec_pot(x_d),1} = strcat('M_int_c.elec_pot', x_d_string);
Scales.SV(Pointer.m_int_c.elec_pot(x_d),1) = Solver.SV_scale.elec_pot;
Scales.dSV(Pointer.m_int_c.elec_pot(x_d),1) = Solver.dSV_scale.electroneutrality;
Units{Pointer.m_int_c.elec_pot(x_d),1} = 'V';
Solver.kinsol.Constraints(Pointer.m_int_c.elec_pot(x_d),1) = 0; % 0 --> none, 1 --> >= 0, 2 --> >0
Grid_size.cathode.state_vars_num = Grid_size.cathode.state_vars_num + 1;
SV_position = SV_position + 1;

% Mass Fractions
Pointer.m_int_c.mass_fracs(x_d) = SV_position;
for species = 1 : Species.oxidizer.num
    Names{Pointer.m_int_c.mass_fracs(x_d) + species - 1,1} = ...
        strcat('M_int_c.mass_fr.' , char(Species.oxidizer.list(species)), x_d_string);
end
Pointer_range.m_int_c = Pointer.m_int_c.mass_fracs(x_d) : ...
    Pointer.m_int_c.mass_fracs(x_d) + Species.oxidizer.num - 1;

```

```

Scales.SV(Pointer_range.m_int_c,1)      = Solver.SV_scale.mass_frac_cathode; % Scale for mass fractions
Scales.dSV(Pointer_range.m_int_c,1)    = Solver.dSV_scale.species_flux_balance_c;
Units(Pointer_range.m_int_c,1)        = {'none'}; % Mass fraction is unitless
Solver.kinsol.Constraints(Pointer_range.m_int_c,1) = 0; % 0 --> none, 1 --> >= 0, 2 --> >0
Grid_size.cathode.state_vars_num      = Grid_size.cathode.state_vars_num + Species.oxidizer.num;
SV_position                           = SV_position + Species.oxidizer.num;

% Pressure
Pointer.m_int_c.press(x_d)             = SV_position;
Names{Pointer.m_int_c.press(x_d),1}   = strcat('M_int_c.pressure', x_d_string);
Scales.SV(Pointer.m_int_c.press(x_d),1) = Solver.SV_scale.press;
Scales.dSV(Pointer.m_int_c.press(x_d),1) = Solver.dSV_scale.mass_flux_balance;
Units{Pointer.m_int_c.press(x_d),1}   = 'Pa';
Solver.kinsol.Constraints(Pointer.m_int_c.press(x_d),1) = 0; % None
Grid_size.cathode.state_vars_num      = Grid_size.cathode.state_vars_num + 1;
SV_position                           = SV_position + 1;

% -----%
% y-Discretization "csln". Cathode electrolyte solution
% -----%

for y_d = 1 : Grid_size.y_d_num

    x_y = strcat('(' , num2str(x_d) , ',' , num2str(y_d) , ')');

    % Electric Potential
    Pointer.csln.elec_pot(x_d,y_d)     = SV_position;
    Names{Pointer.csln.elec_pot(x_d,y_d),1} = strcat('Csln.elec_pot',x_y);
    Scales.SV(Pointer.csln.elec_pot(x_d,y_d),1) = Solver.SV_scale.elec_pot;
    Scales.dSV(Pointer.csln.elec_pot(x_d,y_d),1) = Solver.dSV_scale.electroneutrality;
    Units{Pointer.csln.elec_pot(x_d,y_d),1} = 'V';
    Solver.kinsol.Constraints(Pointer.csln.elec_pot(x_d,y_d),1) = 0;
    Grid_size.cathode.state_vars_num      = Grid_size.cathode.state_vars_num + 1;
    SV_position                           = SV_position + 1;

    % Mass Fractions

```

```

Pointer.csln.mass_fracs(x_d,y_d) = SV_position;
for species = 1 : Species.oxidizer.num
    Names{Pointer.csln.mass_fracs(x_d,y_d) + species - 1,1} = ...
        strcat('Csln.mass_fr.' , char(Species.oxidizer.list(species)), x_y );
end
Pointer_range.csln = Pointer.csln.mass_fracs(x_d,y_d) : ...
    Pointer.csln.mass_fracs(x_d,y_d) + Species.oxidizer.num - 1;
Scales.SV(Pointer_range.csln,1) = Solver.SV_scale.mass_frac_cathode; % Scale for mass fractions
Scales.dSV(Pointer_range.csln,1) = Solver.dSV_scale.species_cons_c;
Units(Pointer_range.csln,1) = {'none'}; % Mass fraction is unitless
Solver.kinsol.Constraints(Pointer_range.csln,1) = 0; % 0 --> none, 1 --> >= 0, 2 --> >0
Grid_size.cathode.state_vars_num = Grid_size.cathode.state_vars_num + Species.oxidizer.num;
SV_position = SV_position + Species.oxidizer.num;

% x-Velocity
Pointer.csln.x_vel(x_d,y_d) = SV_position;
Names{Pointer.csln.x_vel(x_d,y_d),1} = strcat('Csln.x_velocity',x_y);
Scales.SV(Pointer.csln.x_vel(x_d,y_d),1) = Solver.SV_scale.x_vel;
Scales.dSV(Pointer.csln.x_vel(x_d,y_d),1) = Solver.dSV_scale.x_momentum_cons;
Units{Pointer.csln.x_vel(x_d,y_d),1} = 'm/s';
Solver.kinsol.Constraints(Pointer.csln.x_vel(x_d,y_d),1) = 0;
Grid_size.cathode.state_vars_num = Grid_size.cathode.state_vars_num + 1;
SV_position = SV_position + 1;

% y-Velocity
Pointer.csln.y_vel(x_d,y_d) = SV_position;
Names{Pointer.csln.y_vel(x_d,y_d),1} = strcat('Csln.y_velocity',x_y);
Scales.SV(Pointer.csln.y_vel(x_d,y_d),1) = Solver.SV_scale.y_vel;
Scales.dSV(Pointer.csln.y_vel(x_d,y_d),1) = Solver.dSV_scale.y_momentum_cons;
Units{Pointer.csln.y_vel(x_d,y_d),1} = 'm/s';
Solver.kinsol.Constraints(Pointer.csln.y_vel(x_d,y_d),1) = 0;
Grid_size.cathode.state_vars_num = Grid_size.cathode.state_vars_num + 1;
SV_position = SV_position + 1;

% Pressure
Pointer.csln.press(x_d,y_d) = SV_position;

```

```

Names{Pointer.csln.press(x_d,y_d),1} = strcat('Csln.pressure',x_y);
Scales.SV(Pointer.csln.press(x_d,y_d),1) = Solver.SV_scale.press;
Scales.dSV(Pointer.csln.press(x_d,y_d),1) = Solver.dSV_scale.mass_cons;
Units{Pointer.csln.press(x_d,y_d),1} = 'Pa';
Solver.kinsol.Constraints(Pointer.csln.press(x_d,y_d),1) = 0;
Grid_size.cathode.state_vars_num = Grid_size.cathode.state_vars_num + 1;
SV_position = SV_position + 1;

end

% -----%
% y-Discretization "c_int". Cathode electrolyte solution
% -----%

% Electric Potential
Pointer.c_int.elec_pot(x_d) = SV_position;
Names{Pointer.c_int.elec_pot(x_d),1} = strcat('C_int.elec_potent', x_d_string);
Scales.SV(Pointer.c_int.elec_pot(x_d),1) = Solver.SV_scale.elec_pot;
Scales.dSV(Pointer.c_int.elec_pot(x_d),1) = Solver.dSV_scale.electroneutrality;
Units{Pointer.c_int.elec_pot(x_d),1} = 'V';
Solver.kinsol.Constraints(Pointer.c_int.elec_pot(x_d),1) = 0; % 0 --> none, 1 --> >= 0, 2 --> >0
Grid_size.cathode.state_vars_num = Grid_size.cathode.state_vars_num + 1;
SV_position = SV_position + 1;

% Mass Fractions
Pointer.c_int.mass_fracs(x_d) = SV_position;
for species = 1 : Species.oxidizer.num
    Names{Pointer.c_int.mass_fracs(x_d) + species - 1,1} = ...
        strcat('C_int.mass_fr.', char(Species.oxidizer.list(species)), x_d_string );
end
Pointer_range.c_int = Pointer.c_int.mass_fracs(x_d) : ...
    Pointer.c_int.mass_fracs(x_d) + Species.oxidizer.num - 1;
Scales.SV(Pointer_range.c_int,1) = Solver.SV_scale.mass_frac_cathode; % Scale for mass fractions
Scales.dSV(Pointer_range.c_int,1) = Solver.dSV_scale.species_flux_balance_c;
Units(Pointer_range.c_int,1) = {'none'}; % Mass fraction is unitless
Solver.kinsol.Constraints(Pointer_range.c_int,1) = 1; % 0 --> none, 1 --> >= 0, 2 --> >0

```

```

Grid_size.cathode.state_vars_num      = Grid_size.cathode.state_vars_num + Species.oxidizer.num;
SV_position                            = SV_position + Species.oxidizer.num;

end

% Sum the number of state variables in the anode, membrane and cathode
Grid_size.state_vars_num = Grid_size.anode.state_vars_num + Grid_size.cathode.state_vars_num;

%% 3. CREATE THE UNSCALED SOLUTION VECTOR AND ALLOCATE A CONTIGUOUS BLOCK OF MEMORY FOR IT

% Allocate space for all of the state variables in an x-discretization, for all x-discretizations, plus
% one for the cell voltage or the cell current, depending on which the user chooses.
SV_initial_unsc = zeros(Grid_size.state_vars_num, 1);

%% 4. POPULATE THE INITIAL SOLUTION VECTOR

if Flags.setup.new_SV_initial

% Change the strategy to be LineSearch, which works much better from the standard guess
Solver.kinsol.strategy = 'LineSearch';

% For the initial condition, we'll assume that the anode flow is the same as the fuel flow and the
% cathode flow is the same as the oxidizer flow. The following loop steps along the channel, one
% x-discretization at a time, filling in the solution vector as it goes. The outer loop steps down the
% channel and the inner loop steps through all of the species. It uses the pointers defined above to
% fill in the state variables for each discretization. For differential variables, these are the
% initial conditions and for algebraic variables they are initial guesses.

disp('Populating the initial state vector...')

% Find the slope with which the electric potential changes from the inlet to the outlet along each
% interface, given the inlet and outlet values specified by the user.
slope_a_int   = ( Initial.a_int_elec_pot_out   - Initial.a_int_elec_pot_in ) /Geometry.channel_length;
slope_m_int_a = ( Initial.m_int_a_elec_pot_out - Initial.m_int_a_elec_pot_in )/Geometry.channel_length;
slope_m_int_c = ( Initial.m_int_c_elec_pot_out - Initial.m_int_c_elec_pot_in )/Geometry.channel_length;
slope_c_int   = ( Initial.c_int_elec_pot_out   - Initial.c_int_elec_pot_in ) /Geometry.channel_length;

```

```

for x_d = 1 : Grid_size.x_d_num

% ELECTRODE INTERFACE CELLS

% Electric Potential
SV_initial_unsc(Pointer.a_int.elec_pot(x_d)) = Initial.a_int_elec_pot_in + ...
    slope_a_int * Geometry.x_d_location(x_d+1);
SV_initial_unsc(Pointer.c_int.elec_pot(x_d)) = Initial.c_int_elec_pot_in + ...
    slope_c_int * Geometry.x_d_location(x_d+1);

% Mass Fractions
SV_initial_unsc(Pointer.a_int.mass_fracs(x_d) + (1:Species.fuel.num) - 1) = ...
    Fuel.Mass_fractions;
SV_initial_unsc(Pointer.c_int.mass_fracs(x_d) + (1:Species.oxidizer.num) - 1) = ...
    Oxidizer.Mass_fractions;

% MEMBRANE INTERFACE CELLS

% Electric Potential
SV_initial_unsc(Pointer.m_int_a.elec_pot(x_d)) = Initial.m_int_a_elec_pot_in + ...
    slope_m_int_a * Geometry.x_d_location(x_d+1);
SV_initial_unsc(Pointer.m_int_c.elec_pot(x_d)) = Initial.m_int_c_elec_pot_in + ...
    slope_m_int_c * Geometry.x_d_location(x_d+1);

% Mass Fractions
SV_initial_unsc(Pointer.m_int_a.mass_fracs(x_d) + (1:Species.fuel.num) - 1) = ...
    Initial.m_int_a_mass_fracs;
SV_initial_unsc(Pointer.m_int_c.mass_fracs(x_d) + (1:Species.oxidizer.num) - 1) = ...
    Initial.m_int_c_mass_fracs;

% BULK SOLUTION CELLS

% x-Velocity... assume all points down the channel have the same fully developed velocity profile as
% at the inlet.
SV_initial_unsc(Pointer.asln.x_vel(x_d,:)) = BC.anode.x_vel_inlet(2:Grid_size.y_d_num+1);

```

```

SV_initial_unsc(Pointer.csln.x_vel(x_d,:)) = BC.cathode.x_vel_inlet(2:Grid_size.y_d_num+1);

% y-Velocity... establish linear gradeints in the bulk with expected direction
SV_initial_unsc(Pointer.asln.y_vel(x_d,:)) = linspace(0, Initial.m_int_a_y_velocity, ...
    Grid_size.y_d_num);
SV_initial_unsc(Pointer.csln.y_vel(x_d,:)) = linspace(0, Initial.m_int_c_y_velocity, ...
    Grid_size.y_d_num);

% Electric potential

% Calculate the slope at which the electric field changes from electrode to membrane in each channel,
% at each point down the channel, to accomodate the changing values of the potential at the electrode
% and membrane interfaces.
anode_slope(x_d) = (SV_initial_unsc(Pointer.m_int_a.elec_pot(x_d)) - ...
    SV_initial_unsc(Pointer.a_int.elec_pot(x_d))) / Geometry.channel_height;
cathode_slope(x_d) = (SV_initial_unsc(Pointer.m_int_c.elec_pot(x_d)) - ...
    SV_initial_unsc(Pointer.c_int.elec_pot(x_d))) / Geometry.channel_height;

% We're already in an x_d loop.... so for each value of x_d, the following loop fills in the values
% of the electric potential in the bulk electrolyte between the electrode and membrane.
for y_d = 1 : Grid_size.y_d_num

    % Electric Potential... establish linear gradients in the bulk with expected direction
    SV_initial_unsc(Pointer.asln.elec_pot(x_d,y_d)) = SV_initial_unsc(Pointer.a_int.elec_pot(x_d)) ...
        + Geometry.y_d_location(y_d+1) * anode_slope(x_d);
    SV_initial_unsc(Pointer.csln.elec_pot(x_d,y_d)) = SV_initial_unsc(Pointer.c_int.elec_pot(x_d)) ...
        + Geometry.y_d_location(y_d+1) * cathode_slope(x_d);

    % Mass Fractions
    SV_initial_unsc(Pointer.asln.mass_fracs(x_d,y_d) + (1:Species.fuel.num) - 1) = ...
        BC.anode.Mass_fractions_inlet(1,1,:);
    SV_initial_unsc(Pointer.csln.mass_fracs(x_d,y_d) + (1:Species.oxidizer.num) - 1) = ...
        BC.cathode.Mass_fractions_inlet(1,1,:);

end

```

```

clear anode_slope cathode_slope

% Pressure

% Establish the x-direction pressure gradient
anode_slope = (BC.anode.press_outlet - Initial.inlet_press) / Geometry.channel_length;
cathode_slope = (BC.cathode.press_outlet - Initial.inlet_press) / Geometry.channel_length;

SV_initial_unsc(Pointer.asln.press(x_d,:)) = Initial.inlet_press + ...
    Geometry.x_d_location(x_d+1) * anode_slope;
SV_initial_unsc(Pointer.csln.press(x_d,:)) = Initial.inlet_press + ...
    Geometry.x_d_location(x_d+1) * cathode_slope;

% Establish the y-direction pressure gradient by superimposing it over the x-direction gradient.
SV_initial_unsc(Pointer.asln.press(x_d,:)) = SV_initial_unsc(Pointer.asln.press(x_d,:)) + ...
    (linspace(Initial.anode_press, Initial.m_int_a.press, Grid_size.y_d_num))';
SV_initial_unsc(Pointer.csln.press(x_d,:)) = SV_initial_unsc(Pointer.csln.press(x_d,:)) + ...
    (linspace(Initial.cathode_press, Initial.m_int_c.press, Grid_size.y_d_num))';

SV_initial_unsc(Pointer.m_int_a.press(x_d)) = ...
    SV_initial_unsc(Pointer.asln.press(x_d,Grid_size.y_d_num-1));
SV_initial_unsc(Pointer.m_int_c.press(x_d)) = ...
    SV_initial_unsc(Pointer.csln.press(x_d,Grid_size.y_d_num-1));
end

% Scale the initial state vector
SV_initial = SV_initial_unsc ./ Scales.SV;

else

% LOAD THE INITIAL SOLUTION VECTOR FROM A FILE SPECIFIED IN DBFC_USER_INPUT

% Store the present boundary conditions and scales in a temporary variable so they are not lost when
% the file is imported
BC_temp = BC;
Scales_temp = Scales;

```

```

% Import the solution vector and some other data from the file
load(Solver.SV_initial_filename, 'SV_steady_state', 'dSV_steady_state', 'Scales', 'BC')

% Change the strategy to be pure Newton search
Solver.kinsol.strategy = 'None';

% Adjust the voltage field in the old solution to be closer to that of the solution we are looking for.
if Flags.setup.adjust_V_prev_soln
    V_cell_old = BC.cathode.elec_pot;
    V_cell_new = Cathode_electric_potential(1);
    SV_initial = FUNC_ADJUST_SOLN_VOLTAGE( SV_steady_state, V_cell_old, V_cell_new, Scales, ...
        Pointer, Geometry, Grid_size );

    SV_initial_unsc = SV_initial .* Scales.SV;
else
    SV_initial = SV_steady_state;
    SV_initial_unsc = SV_initial .* Scales.SV;
end

% Recover the boundary conditions and scales for this run.
BC = BC_temp;
Scales = Scales_temp;

end

%% 5. RESET SCALES, IF DESIRED

if Flags.setup.rescale_init_guess % Adjust Scales.SV and the intial guess so the initial guess is all 1s
    State_initial = SV_initial .* Scales.SV;  Scales.SV = State_initial;
    SV_initial = ones(length(SV_initial),1);
    SV_initial_unsc = SV_initial .* Scales.SV;  % For printing the unscaled values below
end

%% 6. INITIALIZE THE GLOBAL VARIABLES STORING THE LAST SOLUTION VECTORS BEFORE THE SOLVER FAILS FOR DEBUG
SV_fail_lg = SV_initial;

```

```

SV_fail    = SV_initial;

%% 7. CLEANUP BY CLEARING SOME VARIABLES THAT ARE NO LONGER NEEDED

% Cell arrays cannot be coded into MEX files, so remove the fields in Model containing species names.
Species.fuel = rmfield(Species.fuel, 'list');
Species.oxidizer = rmfield(Species.oxidizer, 'list');
Species.membrane = rmfield(Species.membrane, 'list');

%% 8. CALL THE MAIN FUNCTION ONCE TO COMPUTE THE INITIAL RESIDUAL VALUES

Jacobian.JAC_pat_exists = 0;  Jacobian.bandwidth_exists = 0;

% Set the cell voltage to be the first value in the list in DBFC_USER_INPUT
BC.cathode.elec_pot = Cathode_electric_potential(1);

% Find out what the residuals values are at the initial guess.  This if statement avoids an error due to
% calling the kinsol statistics function before it is initialized, which would happen if we called
% DBFC_FUNCTION before initializing kinsol with Solver.kinsol.display_iter = 1.

if Solver.kinsol.display_iter == 1
    Solver.kinsol.display_iter = 0;    function_start_time = toc;
    dSV_initial = DBFC_FUNCTION(SV_initial);
    Solver.kinsol.display_iter = 1;
else
    function_start_time = toc;
    dSV_initial = DBFC_FUNCTION(SV_initial);
end

function_end_time = toc;
disp(['The function evaluation took: ' num2str(function_end_time - function_start_time) 'sec.'])

%% 9. RESCALE THE INITIAL RESIDUALS, IF DESIRED

if Flags.setup.rescale_init_resid
    Residuals_initial = dSV_initial ./ Scales.dSV;

```

```

Residuals_initial(Residuals_initial==0) = eps; % For initial residuals of zero...
Scales.dSV = Residuals_initial.^-1;
end

%% 10. DISPLAY THE INITIAL SYSTEM STATE

if Flags.setup.display_initial_state

disp(' ')
disp('The initial state of the system is ... ')
disp(' ')

l_i = 0;

for x_d = 1 : Grid_size.x_d_num

disp(['x-Discretization ' num2str(x_d)])
disp('-----')
fprintf('%-8s %-26s %-8s %-18s %-22s %-13s\n', 'Index', 'State Var Name', 'Units', ...
        'Initial Val.', 'Init. Scaled Val.', 'Init. Resids.')
disp('-----')

for y_d = 1 : Grid_size.anode.state_vars_num / Grid_size.x_d_num;

l_i = l_i + 1;
fprintf('%-7u\t %-24s\t %-5s\t %-16E\t %-16f\t %-13E\n' , l_i, char(Names(l_i)), ...
        char(Units(l_i)), SV_initial_unsc(l_i), SV_initial(l_i), dSV_initial(l_i) )

end

disp('-----Membrane-----')

for y_d = 1 : Grid_size.cathode.state_vars_num / Grid_size.x_d_num;

l_i = l_i + 1;
fprintf('%-7u\t %-24s\t %-5s\t %-16E\t %-16f\t %-13E\n' , l_i, char(Names(l_i)), ...

```

```

        char(Units(l_i)), SV_initial_unsc(l_i), SV_initial(l_i), dSV_initial(l_i) )

    end

end

disp(' ')

end

disp(['There are ' num2str(Grid_size.state_vars_num) ' state variables total.'])

%% 8. CREATE OR LOAD THE JACOBIAN PATTERN AND SOME OF ITS CHARACTERISTICS

% Initially, no Jacobian pattern exists
Solver.Jacobian.JAC_pat_exists = 0;    Solver.Jacobian.bandwidth_exists = 0;

% Start a clock for Jacobian calculations
jacobian_start_time = toc;

% Create a variable that is very useful in this cell
length_SV_initial = length(SV_initial);

% Either generate the Jacobian pattern, upper bandwidth and lower bandwidth using one of several methods,
% or load the Jacobian pattern from a file.
switch Solver.Jacobian.Jpattern_flag

    case 'load' % Load the Jacobian pattern from a file, with the filename specified in the input file

        disp('Loading the Jacobian pattern from file...');
        load(j_pattern_filename, 'Jacobian_pattern');
        Model.JAC_pattern = Jacobian_pattern; clear Jacobian_pattern
        %[Model.JAC_upper_bandwidth, Model.JAC_lower_bandwidth] = FUNC_MATRIX_BANDWIDTH(Model.JAC_pattern);
        Jacobian.JAC_pat_exists = 1; % Flags.bandwidth_exists = 1;
        time.JAC_pattern_construction = toc - jacobian_start_time;
        disp(['Jacobian pattern is loaded. Load time was ' num2str(time.JAC_pattern_construction) 's.']);
    end
end

```

```

case 'random' % Calculate the Jacobian pattern using repeated random perturbations to the function

disp('Generating a new Jacobian pattern...');
Model.JAC_pattern = FUNC_JPAT_RANDOM_ASSEMBLE(length_SV_initial);
[Model.JAC_upper_bandwidth, Model.JAC_lower_bandwidth] = FUNC_MATRIX_BANDWIDTH(Model.JAC_pattern);
Jacobian.JAC_pat_exists = 1; Jacobian.bandwidth_exists = 1;
time.JAC_pattern_construction = toc - jacobian_start_time;
disp(' ');
disp(['Jacobian pattern is complete. Generating time was ' ...
    num2str(time.JAC_pattern_construction) 's.']);

case 'analytic' % Calculate the Jacobian pattern using knowledge of the problem structure

disp('Constructing analytical Jacobian pattern...');

% Calculate the Jacobian pattern
Model.JAC_pattern = FUNC_JPAT_ANALYTIC(length_SV_initial, Species, Grid_size);

% Find the upper and lower bandwidths of the Jacobian pattern for
% the banded solver
[Model.JAC_upper_bandwidth, Model.JAC_lower_bandwidth] = FUNC_MATRIX_BANDWIDTH(Model.JAC_pattern);

% Set flag indicating that the Jacobian pattern exists for future calls to the Jacobian calculation
% functions
Jacobian.JAC_pat_exists = 1; Jacobian.bandwidth_exists = 1;

% Calculate and display the time to generate the Jacobian and Jacobian pattern
time.JAC_pattern_construction = toc - jacobian_start_time;

% Display completion message.
disp(['Jacobian pattern is complete. Construction time was ' ...
    num2str(time.JAC_pattern_construction) 's.'])

% Calculate the Jacobian pattern by calculating the Jacobian at the initial guess, then finding the
% non-zero elements

```

```

case 'empirical'

disp('Constructing empirical Jacobian pattern...');

% Set the Jacobian function to return a sparse matrix
Solver.Jacobian.rtrn_dense_JAC = 0;

% Calculate the Jacobian at the initial guess
[Jacobian_sparse, flag] = FUNC_JACOBIAN_NUMJAC(SV_initial, dSV_initial);
%[Jacobian_sparse, flag] = FUNC_JACOBIAN_STROMAN(SV_initial, dSV_initial);

% Find the sparsity pattern of the Jacobian and store it as the Jacobian pattern
Solver.Jacobian.JAC_pattern = spones(Jacobian_sparse);

% Find the upper and lower bandwidths of the sparse Jacobian for the banded solver
[Solver.Jacobian.upper_bandwidth, Solver.Jacobian.lower_bandwidth] = ...
    FUNC_MATRIX_BANDWIDTH(Jacobian_sparse);

% Save the Jacobian and Jacobian pattern to files for reuse later
Jacobian_pattern_temp = Solver.Jacobian.JAC_pattern;
save(strcat('Jacobian_pattern_xd',num2str(Grid_size.x_d_num), '_yd', ...
    num2str(Grid_size.y_d_num),'.mat'), 'Jacobian_pattern_temp');
save(strcat('Jacobian_xd',num2str(Grid_size.x_d_num), '_yd', ...
    num2str(Grid_size.y_d_num),'.mat'), 'Jacobian_sparse');
clear Jacobian_pattern_temp Jacobian_sparse % Don't need either of these... free up some memory

% Set flag indicating Jacobian pattern exists for future calls to the Jacobian calculation functions
Solver.Jacobian.JAC_pat_exists = 1; Solver.Jacobian.bandwidth_exists = 1;

% Calculate and display the time to generate the Jacobian and Jacobian pattern
time.JAC_pattern_construction = toc - jacobian_start_time;

% Display completion message.
disp(['Jacobian pattern complete. Construction time: ' num2str(time.JAC_pattern_construction) 's.'])

```

```

case 'ones' % Jacobian pattern is all ones... calculate the whole Jacobian on each iteration

Solver.Jacobian.JAC_pattern = ones(length_SV_initial);
Solver.Jacobian.JAC_pat_exists = 1;
time.JAC_pattern_construction = toc - jacobian_start_time;
disp(['Created Jacobian pattern of all ones. Construction time: ' ...
    num2str(time.JAC_pattern_construction) 's.']);

% Jacobian pattern is filled area between upper and lower bandwidths specified by the user
case 'specified'

disp('Constructing Jacobian pattern with user specified bandwidths...');

Solver.Jacobian.JAC_pattern = sparse([], [], [], length_SV_initial, length_SV_initial, ...
    (Solver.Jacobian.lower_bandwidth+Solver.Jacobian.upper_bandwidth) * length_SV_initial);

for row_index = 1:length(SV_initial);

    lower_bound = max(row_index - Solver.Jacobian.lower_bandwidth, 1);
    upper_bound = min(row_index + Solver.Jacobian.bandwidth, length(SV_initial));

    Solver.Jacobian.JAC_pattern(row_index, lower_bound:upper_bound) = 1;

end

% Calculate and display the time to generate the Jacobian and Jacobian pattern
time.JAC_pattern_construction = toc - jacobian_start_time;

% Display completion message.
disp(['Jacobian pattern complete. Construction time: ' num2str(time.JAC_pattern_construction) 's.'])

case 'none' % No Jacobian pattern is requested
    disp('No Jacobian pattern requested or generated')
otherwise % Error handling
    error('Unexpected source for the Jacobian pattern.')

```

```

end

% Set the Jacobian matrix bandwidth if supplied by the user

if Jacobian.bandwidth_exists
    disp('Upper and lower Jacobian bandwidths calculated from Jacobian or Jacobian pattern')
else if strcmp(Solver.Jacobian.JAC_bandwidth_flag, 'user')
    Solver.Jacobian.JAC_upper_bandwidth = Solver.Jacobian.upper_bandwidth;
    Solver.Jacobian.JAC_lower_bandwidth = Solver.Jacobian.lower_bandwidth;
    disp('Using user supplied upper and lower Jacobian bandwidths')
    end

end

%% 8. CLEAR UNNECESSARY VARIABLES FROM THE WORKSPACE AND DISPLAY STATUS MESSAGE

clear x_d y_d x_y x_d_string loop_index species_index i j loop_index l_i r_i s_i column_range_high ...
    column_range_low Pointer_range position_index_anode position_index_cathode species position_offset ...
    previous_SV_initial_path conc_f conc_r SV_position jacobian_start_time function_end_time ....
    function_start_time

%% SUMMARY: DBFC_KINSOL
% Purpose: Call kinsol to solve the model.
% Author: Rick Stroman

%% NOTES
%
% (1) Results are stored in SV_steady_state (the solution vector), and dSV_steady_state (the time
% derivatives and residuals) which is the system state at infinite time, given constant user input
% parameters.

%% SETUP THE SOLVER

global num_nonlin_iter Model Flags Solver

```

```

switch Solver.kinsol.linear_solver

    case 'Dense'

        Flags.return_dense_Jacobian = 1;

        options = KINSetOptions(...
            'Verbose',          Solver.kinsol.verbose, ...
            'FuncNormTol',     Solver.kinsol.func_norm_tol, ...
            'ScaledStepTol',   Solver.kinsol.scaled_step_tol, ...
            'LinearSolver',     Solver.kinsol.linear_solver, ...
            'JacobianFn',      'FUNC_CALC_JACOBIAN');

    case 'Band'

        options = KINSetOptions(...
            'Verbose',          Solver.kinsol.verbose, ...
            'FuncNormTol',     Solver.kinsol.func_norm_tol, ...
            'ScaledStepTol',   Solver.kinsol.scaled_step_tol, ...
            'LinearSolver',     Solver.kinsol.linear_solver, ...
            'MaxNewtonStep',   Solver.kinsol.MaxNewtonStep, ...
            'LowerBwidth',     Solver.Jacobian.lower_bandwidth, ...
            'UpperBwidth',     Solver.Jacobian.upper_bandwidth, ...
            'Constraints',     Solver.kinsol.Constraints, ...
            'MaxNumBetaFails', Solver.kinsol.MaxNumBetaFails, ...
            'MaxNewtonStep',   Solver.kinsol.MaxNewtonStep, ...
            'MaxNumSetups',    Solver.kinsol.MaxNumSetups, ...
            'MaxNumIter',      Solver.kinsol.MaxNumIter);

end

% Globalization strategy
strategy = Solver.kinsol.strategy; % Strategy for the linear solver

% Set number of equations, and scaling on the solution variables and function

```

```

num_eqns = length(SV_initial);
yscale   = ones(num_eqns,1); % If ones, the model uses Scale.SV to scale the input
fscale   = ones(num_eqns,1); % If ones, the model uses Scale.dSV to scale the residuals

% Initialize the solver
KINInit(@DBFC_FUNCTION, num_eqns, options);

disp(' ')
disp(strcat('Steady-state simulation using KINSOL and the linear solver solver...'));

%% SOLVE THE MODEL

% Set counter value for outputting solver status during solve process
num_nonlin_iter = 0;

% Header for residuals printed from function
disp(' ');
fprintf('%-15s  %-18s %-17s %-12s \n', 'Non-lin. Iter.', 'Func Evals', 'Resid 2-Norm', 'Time (min)')
disp('-----')

Init_resid_norm = norm(DBFC_FUNCTION(SV_initial));

fprintf('%-15.0f  %-20.3e %-20.3e %-20.3f \n', 0, 1, Init_resid_norm, toc/60)

% Call the solver
[termination_status, SV_steady_state] = KINSol(SV_initial, strategy, yscale, fscale);
dSV_steady_state                       = DBFC_FUNCTION(SV_steady_state);

% HANDLE AND REPORT THE SOLVER OUTPUT

disp(' ')

switch termination_status
    case 0
        disp('KINSol succeeded')

```

```

    case 1
        disp('The initial y0 already satisfies the stopping criterion given above')
    case 2
        disp('Stopping tolerance on scaled step length satisfied')
    case -1
        disp('An error occurred (see printed error message)')
end

KINSOL_status_structure = KINGetStats;

disp(' ')
disp('Status output from KINSol:')
disp('-----')
disp(KINSOL_status_structure)
disp('-----')
disp(' ')

%% CLEANUP

KINFree; % Release the memory that was allocated for KinSol

function [d_SV, error_flag ] = DBFC_FUNCTION(SV)

%% SUMMARY: DBFC_FUNCTION

% Purpose: This function accepts a guess at the solution vector and computes the time rates of change for
% differential variables and the residuals for algebraic variables.
% Author: Rick Stroman

%% NOTES
%
% (1) SV contains either the initial state of the system, or the state at the conclusion of the
% previous time step.

```

```

%
% (2) d_SV contains the time rate of change of each differential state variable and the residual for each
% algebraic variable, at the present time step. The purpose of this function is to compute d_SV for the
% solver... when a steady state solver is used, d_SV is driven to zero, and when a transient (ODE) solver
% is used, the values of d_SV associated with differential equations are integrated to find the system
% state at each time step and the values associated with algebraic equations are driven to zero.
%
% (3) Acronyms
% SCB: Scalar Cell Boundary
% VCB: Velocity Cell Boundary
% ASLN: Anode SoLutioN
% CSLN: Cathode SoLutioN
% SV: State Vector

%% 1. DECLARE MODEL-WIDE GLOBAL VARIABLES

global Properties Scales Geometry Pointer Flags Constants BC Arxn Crxn SV_fail SV_fail_lg ...
    num_nonlin_iter Names Solver total_cathode_current Grid_size Species

% These lines are used for debug... they are global variables I can look at after the code bombs.
SV_fail_lg = SV_fail; % Value of SV from the last good iteration
SV_fail = SV; % Value of SV at which the code bombed

%% 2. PREALLOCATE MEMORY FOR RESIDUALS

% % Allocate memory for the residuals and time derivatives, and initialize them
d_SV = zeros(length(SV),1);

%% 3. READ THE PRESENT SYSTEM STATE OUT OF SV

[ State.A_int, State.Asln, State.M_int_a, State.M_int_c, State.Csln, State.C_int, State.C ] = ...
    FUNC_READ_SOLUTION_VECTOR(SV, Scales, Pointer, Grid_size, Species, BC);

% Adjust electric potential guesses that effect reaction rates if they are large enough to cause
% overflow problems in the reaction rate estimation function.
if any(abs(State.A_int.elec_pot) > 2)

```

```

    State.A_int.elec_pot = 4 * sign(State.A_int.elec_pot) + ...
        1e-2 * sign(State.A_int.elec_pot) .* (abs(State.A_int.elec_pot)-8);
end

if any(abs(State.C_int.elec_pot) > 4)
    State.C_int.elec_pot = 4 * sign(State.C_int.elec_pot) + ...
        1e-2 * sign(State.C_int.elec_pot) .* (abs(State.C_int.elec_pot)-8);
end

if any(abs(State.C.elec_pot) > 4)
    State.C.elec_pot = 4 * sign(State.C.elec_pot) + ...
        1e-2 * sign(State.C.elec_pot) .* (abs(State.C.elec_pot)-8);
end

%% 4. CREATE GHOST CELLS AROUND THE BULK ELECTROLYTE SOLUTION

% The output of each function below is overwrites the original data in the specified structure.

[State.Asln] = FUNC_CREATE_GHOST_CELLS(State.Asln, Grid_size);
[State.Csln] = FUNC_CREATE_GHOST_CELLS(State.Csln, Grid_size);

%% 5. ASSIGN BOUNDARY CONDITIONS AND INTERFACE VALUES TO GHOST CELLS

% The output of this function overwrites the original data in the specified structure.
[State.Asln, State.Csln] = FUNC_ASSIGN_BCS_AND_INTERFACES(State.A_int, State.Asln, State.M_int_a, ...
    State.M_int_c, State.Csln, State.C_int, Grid_size, BC);

%% 6. CALCULATE THE SOLUTION PROPERTIES AND FLUXES

if Flags.setup.MEX

    [State, Yfluxes, Xfluxes, Membrane_mass_flux, SCB, VCB] = ...
        FUNC_PROP_AND_FLUX_CALCS_mex(State, Pointer, Flags, Geometry, Properties, Constants);

    Yfluxes.mass.arxn = FUNC_ANODE_REACTION_FLUXES(BC.anode.elec_pot, State.A_int, Arxn, ...
        Properties.anode, Grid_size, Species.fuel, Constants);

```

```

Yfluxes.mass.crxn = FUNC_REACTION_FLUXES(State.C.elec_pot, State.C_int, Crxn, ...
    Properties.cathode, Grid_size, Species.oxidizer, Constants);

else

[State, Yfluxes, Xfluxes, Membrane_mass_flux, SCB, VCB] = ...
    FUNC_PROP_AND_FLUX_CALC(S, Pointer, Flags, Geometry, Grid_size, Species, Properties, ...
        Constants);

Yfluxes.mass.arxn = FUNC_ANODE_REACTION_FLUXES(BC.anode.elec_pot, State.A_int, Arxn, ...
    Properties.anode, Grid_size, Species.fuel, Constants);
Yfluxes.mass.crxn = FUNC_REACTION_FLUXES(State.C.elec_pot, State.C_int, Crxn, ...
    Properties.cathode, Grid_size, Species.oxidizer, Constants);

end

anode_species_range = 1 : Species.fuel.num;
cathode_species_range = 1 : Species.oxidizer.num;

for x_d = 1 : Grid_size.x_d_num

    % Compute reaction mole fluxes [kmol/(m^2 s)]
    Yfluxes.mole.arxn(x_d, anode_species_range) = ...
        Yfluxes.mass.arxn(x_d, 1:length(Properties.anode.one_over_molar_mass)) ...
        .* Properties.anode.one_over_molar_mass';
    Yfluxes.mole.crxn(x_d, cathode_species_range) = ...
        Yfluxes.mass.crxn(x_d, 1:length(Properties.cathode.one_over_molar_mass)) ...
        .* Properties.cathode.one_over_molar_mass';

    % Compute the reaction charge fluxes [C/(m^2 s)]
    Yfluxes.charge.arxn(x_d, 1) = ...
        Constants.faraday * Properties.anode.Electric_charge_T * Yfluxes.mole.arxn(x_d, :)' ;
    Yfluxes.charge.crxn(x_d, 1) = ...
        Constants.faraday * Properties.cathode.Electric_charge_T * Yfluxes.mole.crxn(x_d, :)' ;

end

```

```

%% 7. CALCULATE THE RESIDUALS

if Flags.setup.MEX

    [d_SV] = FUNC_RESIDUALS_mex(State, Yfluxes, Xfluxes, Membrane_mass_flux, SCB, VCB, ...
        Pointer, Flags, Geometry, Properties, Constants); %Grid_size, Species,

else

    [d_SV] = FUNC_RESIDUALS(State, Yfluxes, Xfluxes, Membrane_mass_flux, SCB, VCB, ...
        Pointer, Flags, Geometry, Grid_size, Species, Properties, Constants);

end

%% 8. CALCULATE THE TOTAL CELL CURRENT

% It isn't strictly necessary to do this here.... the current density at each electrode is calculated
% again in postprocessing. This was used at one point for some debugging.

% Initialize the cathode current at each x-discretization
cathode_current = zeros(Grid_size.x_d_num,1);

% CALCULATE THE LOCAL CURRENT DENSITY AT THE CATHODE

for x_d = 2 : Grid_size.x_d_num + 1 % Run through all real cells between the inlet and outlet in the DCS

    x_r = x_d - 1; % Corresponding position in the RCS
    cathode_current(x_r) = Geometry.y_flux_area(x_d) * Yfluxes.charge.crxn(x_r);

end

total_cathode_current = sum(cathode_current);

%% 9. SCALE THE RESIDUALS BEFORE RETURNING THEM FROM THE FUNCTION

```

```

d_SV = d_SV .* Scales.dSV;

%% 10. ERROR HANDLING

% The variable error_flag is only used by the Sundials solvers...

% Check residuals for NaN or Inf...
if isfinite(d_SV) % If the values are all real and finite, tell the solver everything is OK

    error_flag = 0;
else
    % If there is residual which is NaN or Inf, throw an error and tell the solver there is an
    % unrecoverable error (error_flag < 0)

    disp(' '); disp('DBFC_FUNCTION has returned one or more NaN or Inf'); disp(' ');
    error_flag = -1;

    % Display which variables are associated with the NaNs
    disp('The following variables are associated with NaN:')
    Names(find(isnan(d_SV)))
    disp(' ')

    % Display which variables are associated with the Infs
    disp('The following variables are associated with Inf:')
    Names(find(isinf(d_SV)))
end

%% 11. KINSOL OUTPUT

% If the solver is Kinsol AND we want to display iterative output
if Solver.kinsol.display_iter

    Solver_statistics = KINGetStats;

```

```

if Solver_statistics.nni > num_nonlin_iter

    % If the number of nonlinear iterations has increased, print the
    % present statistics to the command window
    num_nonlin_iter = Solver_statistics.nni;
    fprintf('%-15.0f  %-20.3e %-20.3e %-20.3f \n', num_nonlin_iter, Solver_statistics.nfe, ...
        Solver_statistics.fnorm, toc/60)

    if Flags.setup.plot_curr_density
        figure(100+num_nonlin_iter);
        plot(Geometry.x_d_location(2:Grid_size.x_d_num+1), Yfluxes.charge.arxn, 'b-o', ...
            Geometry.x_d_location(2:Grid_size.x_d_num+1), -Yfluxes.charge.crxn, 'r-o')
        title('Local anode and cathode current densities for present iteration')
        xlabel('Distance from inlet [m]')
        ylabel('Local Current Density [A/m^2]')
        legend('Anode Current Density', 'Cathode Current Density')
    end

end

end

end

function [ A_int, Asln, M_int_a, M_int_c, Csln, C_int, C ] = ...
    FUNC_READ_SOLUTION_VECTOR(SV, Scales, Pointer, Grid_size, Species, BC)

%% SUMMARY: FUNC_READ_SOLUTION_VECTOR
% Purpose: This function reads the values out of a solution vector and assigns them to variables
% with nicer names which are indexed by x- and y-discretization, not position in the vector.
% Author: Rick Stroman

%% NOTES

% (1) This code was put in a separate script so that multiple functions and scripts can call it when

```

```

% they want to access the solution vector. When called by a function (such as DBFC_FUNCTION),
% the results of this script disappear with all the rest of that function once it is complete.
%
% (2) For this script to work properly, the solution vector must be called SV. Any solution vector with
% the correct format can be used, including SV_initial, SV (an intermediate solution),
% SV_steady_state or one timestep in SM_transient. Scaling is included, so the new variables
% should have the correct (real world) units.
%
% (3) In each variable produced by this script, the x-discretizations appear in columns (inlet at
% top, outlet at bottom), y-discretizations appear in rows (anode at left, cathode at right) and
% species are listed in the third dimension.
%
% For interfaces:
%
% x 1, species 1      x 1, species 2      x 1, species 3
% x 2, species 1      x 2, species 2      x 2, species 3
% x 3, species 1      x 3, species 2      x 3, species 3
% x 4, species 1      x 4, species 2      x 4, species 3
%
% For electrolyte solution:
%
%           species 3      species 3      species 2
%           species 2      species 2      species 2
%           species 1      species 1      species 1
% x 1, y 1      x 1, y 2      x 1, y 3
% x 2, y 1      x 2, y 2      x 2, y 3
% x 3, y 1      x 3, y 2      x 3, y 3
% x 4, y 1      x 4, y 2      x 4, y 3
%
% (4) The some variables are extracted without a loop, some with only the x index and some with both
% x and y indexes. The amount of looping has been minimized as much as possible to speed up the
% code.
%% 1. INITIALIZE VARIABLES TO SPEED UP LOOPS AND ENSURE THE CORRECT DIMENSIONS

```

```
% Without this step, there would be some ambiguity in the variables which get updated outside of loops if
% either channel dimension is set to 1... by initializing them, we avoid that problem.
```

```
% Interfaces
```

```
A_int.elec_pot = zeros(Grid_size.x_d_num, 1);
C_int.elec_pot = zeros(Grid_size.x_d_num, 1);
```

```
M_int_a.elec_pot = zeros(Grid_size.x_d_num, 1);
M_int_c.elec_pot = zeros(Grid_size.x_d_num, 1);
```

```
A_int.Mass_fracs = zeros(Grid_size.x_d_num, Species.fuel.num);
C_int.Mass_fracs = zeros(Grid_size.x_d_num, Species.oxidizer.num);
```

```
M_int_a.Mass_fracs = zeros(Grid_size.x_d_num, Species.fuel.num);
M_int_c.Mass_fracs = zeros(Grid_size.x_d_num, Species.oxidizer.num);
```

```
M_int_a.press = zeros(Grid_size.x_d_num, 1);
M_int_c.press = zeros(Grid_size.x_d_num, 1);
```

```
% M_int_a.y_vel = zeros(Geometry.x_d_num, 1);
% M_int_c.y_vel = zeros(Geometry.x_d_num, 1);
```

```
% Electrolyte solution
```

```
Asln.elec_pot = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
Csln.elec_pot = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
```

```
Asln.Mass_fracs = zeros(Grid_size.x_d_num, Grid_size.y_d_num, Species.fuel.num);
Csln.Mass_fracs = zeros(Grid_size.x_d_num, Grid_size.y_d_num, Species.oxidizer.num);
```

```
Asln.x_vel = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
Csln.x_vel = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
```

```
Asln.y_vel = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
Csln.y_vel = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
```

```

Asln.press = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
Csln.press = zeros(Grid_size.x_d_num, Grid_size.y_d_num);

%% 2. EXTRACT STATE VARIABLES FROM THE SOLUTION VECTOR AND STORE THEM IN LOCAL VARIABLES

% Unscale the solution vector
SV_unsc = SV .* Scales.SV;

% ELECTRODE INTERFACE CELLS
% Electric Potential
A_int.elec_pot(:,1) = SV_unsc(Pointer.a_int.elec_pot(:));
C_int.elec_pot(:,1) = SV_unsc(Pointer.c_int.elec_pot(:));

% MEMBRANE INTERFACE CELLS

% Electric Potential
M_int_a.elec_pot(:,1) = SV_unsc(Pointer.m_int_a.elec_pot(:));
M_int_c.elec_pot(:,1) = SV_unsc(Pointer.m_int_c.elec_pot(:));

M_int_a.press(:,1) = SV_unsc(Pointer.m_int_a.press(:));
M_int_c.press(:,1) = SV_unsc(Pointer.m_int_c.press(:));

% M_int_a.y_vel(:,1) = SV_unsc(Pointer.m_int_a.y_vel(:));
% M_int_c.y_vel(:,1) = SV_unsc(Pointer.m_int_c.y_vel(:));

% ELECTROLYTE SOLUTION CELLS

% Electric Potential
Asln.elec_pot(:, :) = SV_unsc(Pointer.asln.elec_pot(:, :));
Csln.elec_pot(:, :) = SV_unsc(Pointer.csln.elec_pot(:, :));

% Velocities
Asln.x_vel(:, :) = SV_unsc(Pointer.asln.x_vel(:, :));
Asln.y_vel(:, :) = SV_unsc(Pointer.asln.y_vel(:, :));

```

```

Csln.x_vel(:, :) = SV_unsc(Pointer.csln.x_vel(:, :));
Csln.y_vel(:, :) = SV_unsc(Pointer.csln.y_vel(:, :));

% Pressures
Asln.press(:, :) = SV_unsc(Pointer.asln.press(:, :));
Csln.press(:, :) = SV_unsc(Pointer.csln.press(:, :));

C.elec_pot = BC.cathode.elec_pot;

% Unfortunately some variables must be updated in loops becuse two " : " operators in one set of
% indexes is ambiguous.
for x_d = 1 : Grid_size.x_d_num

    % ELECTRODE INTERFACE CELLS

    % Mass Fractions
    A_int.Mass_fractions(x_d, :) = SV_unsc(Pointer.a_int.mass_fractions(x_d) + (1:Species.fuel.num) - 1);
    C_int.Mass_fractions(x_d, :) = SV_unsc(Pointer.c_int.mass_fractions(x_d) + (1:Species.oxidizer.num) - 1);

    % MEMBRANE INTERFACE CELLS

    % Mass Fractions
    M_int_a.Mass_fractions(x_d, :) = SV_unsc(Pointer.m_int_a.mass_fractions(x_d) + (1:Species.fuel.num) - 1);
    M_int_c.Mass_fractions(x_d, :) = SV_unsc(Pointer.m_int_c.mass_fractions(x_d) + (1:Species.oxidizer.num) - 1);

    % ELECTROLYTE SOLUTION CELLS

    for y_d = 1 : Grid_size.y_d_num

        % Mass Fractions
        Asln.Mass_fractions(x_d, y_d, :) = SV_unsc(Pointer.asln.mass_fractions(x_d, y_d) : ...
            (Pointer.asln.mass_fractions(x_d, y_d) + Species.fuel.num) - 1);
        Csln.Mass_fractions(x_d, y_d, :) = SV_unsc(Pointer.csln.mass_fractions(x_d, y_d) : ...
            (Pointer.csln.mass_fractions(x_d, y_d) + Species.oxidizer.num) - 1);
    end
end

```

```

    end

end

end

function [Sln] = FUNC_CREATE_GHOST_CELLS( Sln, Grid_size )

%% SUMMARY: CREATE_GHOST_CELLS
% Purpose: This script shifts the matrices containing state variables in the electrolyte solution by
% one cell in the x- and y-directions, then adds a ring of ghost cells around the perimeter wherever
% boundary conditions will be specified. Cells in the ring are all assigned the value
% placeholder_value.
% Author: Rick Stroman
% Date: 15 February 2011

%% NOTES

% 1. Ghost cells around the species mass fractions matrices are assigned values, but there are only
% boundary conditions at the inlets. At other locations the ghost cells are simply making it
% possible to calculate solution properties in the ghost cells, which are used to calculate some
% fluxes which form boundary conditions.

% 2. Ghost cells are initially assigned the value initial_value. A value is chosen which is not
% used in a boundary condition or state variable to make it easy to evaluate whether or not the
% boundary conditions and interface values have been applied correctly when looking at the matrices.

%% 1. SETUP THE NECESSARY VARIABLES

channel_x_range = 2:Grid_size.x_d_num+1;
channel_y_range = 2:Grid_size.y_d_num+1;
x_d_ub = Grid_size.x_d_num + 2; % Total num cells is the number of real cells plus one ghost at each end
y_d_ub = Grid_size.y_d_num + 2; % Total num cells is the number of real cells plus one ghost at each end

```

```

placeholder_value = 0;

%% 2. CREATE ELECTRIC POTENTIAL GHOST CELLS

% We need ghost cells for electric potential along each boundary so we can evaluate the migration fluxes
% and electric body forces. Both are written using center differences, so to evaluate these quantities
% in the boundary cells, we need a ghost cell to compute the derivatives. Alternatively, one can look at
% what the model is doing as determining the time rate of change in the amount of species k in a
% differential volume due to migration, which involves a second derivative (analogously to Fick's Second
% Law for diffusion) and hence two boundary conditions are required in each direction.

% Shift
Sln.elec_pot(channel_x_range, channel_y_range) = Sln.elec_pot;

% Create ghost cells
Sln.elec_pot(1,:) = placeholder_value; % Inlet
Sln.elec_pot(x_d_ub,:) = placeholder_value; % Outlet
Sln.elec_pot(:,1) = placeholder_value; % (an - sol interface) or (mem-cat solution interface)
Sln.elec_pot(:, y_d_ub) = placeholder_value; % (anode sol - mem interface) or (sol - cat interface)

%% 3. CREATE MASS FRACTION GHOST CELLS

% We need ghost cells for mass fraction along each boundary so we can evaluate the diffusion fluxes. The
% diffusion flux is written using a center difference, so calculating the diffusion flux in boundary
% cells requires a ghost cell for the derivatives. Alternatively, one can look at what the model is
% ultimately doing: evaluating Fick's Second Law, which gives the time rate of change of species k in a
% differential volume and contains a second derivative, hence we need two boundary conditions in each
% direction.

% Shift
Sln.Mass_fracs(channel_x_range, channel_y_range, :) = Sln.Mass_fracs;

% Create ghost cells
Sln.Mass_fracs(1,::) = placeholder_value; % Inlet
Sln.Mass_fracs(x_d_ub,::) = placeholder_value; % Outlet
Sln.Mass_fracs(:,1,:) = placeholder_value; % (an - sol interface) or (mem-cat solution interface)

```

```

Sln.Mass_fracs(:, y_d_ub, :) = placeholder_value; % (anode sol - mem interface) or (sol - cat interface)

%% 4. CREATE x-VELOCITY GHOST CELLS

% The x-velocity has a first derivative in the x-direction for the advection terms, and a second
% derivative in the y-direction for the shear stress terms. Hence we need only one boundary condition in
% the x-direction and two in the y-direction. We've chosen to specify the velocity at the inlet and the
% electrode and membrane interfaces so they are Dirichlet boundary conditions. The inlet was chosen over
% the outlet so the equations can be formulated as upwind differences, which is more stable for advection
% than center or downwind differencing.

% Shift
Sln.x_vel(channel_x_range, channel_y_range) = Sln.x_vel;

% Create ghost cells
Sln.x_vel(1,:) = placeholder_value; % Inlet
Sln.x_vel(x_d_ub,:) = placeholder_value; % Outlet
Sln.x_vel(:,1) = placeholder_value; % (an - sol interface) or (mem-cat solution interface)
Sln.x_vel(:,y_d_ub) = placeholder_value; % (anode sol - mem interface) or (sol - cat interface)

%% 5. CREATE y-VELOCITY GHOST CELLS

% The y-velocity has a first derivative in the y-direction for the advection terms, and a second
% derivative in the x-direction for the shear stress terms. Hence we need only one boundary condition in
% the y-direction and two in the x-direction. We've chosen to specify the velocity at the inlets,
% outlets and the (anode-solution interface) on the anode side or the (membrane-cathode solution
% interface) on the cathode side, so they are Dirichlet boundary conditions. The (anode-solution
% interface) and (membrane-cathode solution interface) were chosen so the advection terms could be
% formulated as upwind differences.

% Shift
Sln.y_vel(channel_x_range, channel_y_range) = Sln.y_vel;

% Create ghost cells
Sln.y_vel(1,:) = placeholder_value; % Inlet
Sln.y_vel(x_d_ub,:) = placeholder_value; % Outlet

```

```

Sln.y_vel(:,1) = placeholder_value; % Electrode interface
% Sln.y_vel(:, y_d_ub) = placeholder_value; % Membrane interface

%% 6. CREATE PRESSURE GHOST CELLS

% There is only a first derivative of pressure in each direction, so we only need one boundary condition.
% We've chosen to specify the pressure at the inlet to be consistent with the upwind differencing
% formulations of the equations. The pressure gradient is specified at both electrodes where the
% y-direction velocity must be zero due to mass conservation... nothing passes through or is stored on
% the surface. Both the electrode and membrane interface are given ghost cells here just to keep the
% code consistent and allow us to use the same ghost cell generating function for both the anode and
% cathode... one set of ghost cells for each channel is ignored in the rest of the code.

% Shift
Sln.press(channel_x_range, channel_y_range) = Sln.press;

% Create ghost cells
Sln.press(1,:) = placeholder_value; % Inlet (but these ghost cells are never used!)
Sln.press(x_d_ub,:) = placeholder_value; % Outlet
Sln.press(:,1) = placeholder_value; % Electrode interface
Sln.press(:, y_d_ub) = placeholder_value; % Membrane interface

end

function [Asln, Csln] = FUNC_ASSIGN_BCS_AND_INTERFACES(A_int, Asln, M_int_a, M_int_c, Csln, C_int, ...
                                                    Grid_size, BC)

%% SUMMARY: FUNC_ASSIGN_BCS_AND_INTERFACES
% Purpose: This function applies the boundary conditions to the model domain and copies state variables
% from the anode, cathode and membrane interfaces into the bulk electrolyte ghost cells.
% Author: Rick Stroman

```

```
%% NOTES
```

```
% 1. Boundary conditions are assigned to ghost cells at the inlet, anode and cathode interfaces.
```

```
% 2. State variables from the anode, cathode and membrane interfaces are copied into the ghost cells  
% around the bulk electrolyte solution to simplify the flux calculations (eliminates special flux  
% equations for the interfaces) and to simplify plotting the results (same reason).
```

```
%% 1. SET UP THE NECESSARY VARIABLES
```

```
x_d_range_chan1 = 2:Grid_size.x_d_num+1; % Range of x_d values excluding ghost cells  
y_d_range_chan1 = 2:Grid_size.y_d_num+1; % Range of y_d values excluding ghost cells  
x_d_ub = Grid_size.x_d_num + 2; % Largest value of x_d (ghost cell at outlet)  
y_d_ub = Grid_size.y_d_num + 2; % Largest value of y_d (ghost cells at anode  
% solution-membrane interface and at the cathode  
% solution-cathode interface)
```

```
%% 2. APPLY DIRICHLET AND NEWMAN BOUNDARY CONDITIONS
```

```
% Flux boundary conditions at the electrodes and membrane are applied in DBFC_FUNCTION when the  
% conservation equations are evaluated. Voltage boundary conditions for the electrodes are also applied  
% in DBFC_FUNCTION when Kirchoff's Law is applied to electrode discretizations.
```

```
% BC's AT INLET
```

```
% Electric field at inlet is zero (Newmann)  $E = d \phi / dx = 0$   
Asln.elec_pot(1,y_d_range_chan1) = Asln.elec_pot(2,y_d_range_chan1);  
Csln.elec_pot(1,y_d_range_chan1) = Csln.elec_pot(2,y_d_range_chan1);
```

```
% Inlet y-velocity - (Newman)
```

```
Asln.y_vel(1,y_d_range_chan1) = Asln.y_vel(2,y_d_range_chan1);  
Csln.y_vel(1,y_d_range_chan1) = Csln.y_vel(2,y_d_range_chan1);
```

```
% Mass fractions in inlet flows (Dirichlet)
```

```
for y_d = y_d_range_chan1  
    Asln.Mass_fracs(1,y_d,:) = BC.anode.Mass_fractions_inlet(1,1,:);  
    Csln.Mass_fracs(1,y_d,:) = BC.cathode.Mass_fractions_inlet(1,1,:);
```

```

end

% Inlet x-velocity - user specified (Dirichlet)
Asln.x_vel(1,y_d_range_chan1) = BC.anode.x_vel_inlet(y_d_range_chan1);
Csln.x_vel(1,y_d_range_chan1) = BC.cathode.x_vel_inlet(y_d_range_chan1);

% BC'S AT OUTLET

% Electric field at outlet is zero (Newmann)  $E = d \phi / dx = 0$ 
Asln.elec_pot(x_d_ub,y_d_range_chan1) = Asln.elec_pot(x_d_ub-1,y_d_range_chan1);
Csln.elec_pot(x_d_ub,y_d_range_chan1) = Csln.elec_pot(x_d_ub-1,y_d_range_chan1);

% Mass fractions in outlet flows... gradient is zero (Newman)
for y_d = y_d_range_chan1
    Asln.Mass_frac(x_d_ub,y_d,:) = Asln.Mass_frac(x_d_ub-1,y_d,:);
    Csln.Mass_frac(x_d_ub,y_d,:) = Csln.Mass_frac(x_d_ub-1,y_d,:);
end

% Exit x-velocity - assume fully developed, i.e.  $dv_x/dx = 0$  (Newman)
Asln.x_vel(x_d_ub,y_d_range_chan1) = Asln.x_vel(x_d_ub-1,y_d_range_chan1);
Csln.x_vel(x_d_ub,y_d_range_chan1) = Csln.x_vel(x_d_ub-1,y_d_range_chan1);

% Exit y-velocity - assume fully developed, i.e.  $dv_y/dx = 0$  (Newman)
Asln.y_vel(x_d_ub,y_d_range_chan1) = Asln.y_vel(x_d_ub-1,y_d_range_chan1);
Csln.y_vel(x_d_ub,y_d_range_chan1) = Csln.y_vel(x_d_ub-1,y_d_range_chan1);

% Outlet pressure (Dirichlet)
Asln.press(x_d_ub,:) = BC.anode.press_outlet;
Csln.press(x_d_ub,:) = BC.cathode.press_outlet;

% BC's AT ELECTRODES AND MEMBRANE

% No-slip condition at electrodes (Dirichlet)
Asln.x_vel(:,1) = BC.anode.x_vel_electrode;
Csln.x_vel(:,1) = BC.cathode.x_vel_electrode;

```

```

% No-slip condition at membrane (Dirichlet)
Asln.x_vel(:,y_d_ub) = BC.anode.x_vel_membrane;
Csln.x_vel(:,y_d_ub) = BC.cathode.x_vel_membrane;

% y-velocity is zero at the electrodes because there is no net mass flux (Dirichlet)
Asln.y_vel(:,1) = BC.anode.y_vel_electrode;
Csln.y_vel(:,1) = BC.cathode.y_vel_electrode;

% No pressure gradient in y-direction at electrodes (Newmann)
Asln.press(x_d_range_chanl,1) = Asln.press(x_d_range_chanl,2);
Csln.press(x_d_range_chanl,1) = Csln.press(x_d_range_chanl,2);

%% 3. ASSIGN INTERFACE VALUES TO BULK SOLUTION SCALARS IN GHOST CELLS

% Note that the interfaces have no values for velocity or pressure, since they are all handled by
% boundary conditions above. Only the electric potential and mass fractions are relevent.

% ELECTRIC POTENTIAL
Asln.elec_pot(x_d_range_chanl,1) = A_int.elec_pot(:);
Asln.elec_pot(x_d_range_chanl,y_d_ub) = M_int_a.elec_pot(:);
Csln.elec_pot(x_d_range_chanl,1) = C_int.elec_pot(:);
Csln.elec_pot(x_d_range_chanl,y_d_ub) = M_int_c.elec_pot(:);

% PRESSURE
Asln.press(x_d_range_chanl,y_d_ub) = M_int_a.press(:);
Csln.press(x_d_range_chanl,y_d_ub) = M_int_c.press(:);

% Need a loop here because more than one index range is ambiguous
for x_d = x_d_range_chanl

% The x_d-1 accounts for the lack of ghost cells in the interface discretizations, so thier
% x-discretization indices are shifted by -1 with respect to those of the bulk solution.

% MASS FRACTIONS
Asln.Mass_fracs(x_d, 1, :) = A_int.Mass_fracs(x_d-1, :);
Asln.Mass_fracs(x_d, y_d_ub, :) = M_int_a.Mass_fracs(x_d-1, :);

```

```

Csln.Mass_fracs(x_d, 1, :)      = C_int.Mass_fracs(x_d-1, :);
Csln.Mass_fracs(x_d, y_d_ub, :) = M_int_c.Mass_fracs(x_d-1, :);

end

end

function [State_out, Yfluxes, Xfluxes, Membrane_mass_flux, SCB, VCB] = ...
    FUNC_PROP_AND_FLUX_CALC(State, Pointer, Flags, Geometry, Grid_size, Species, Properties, ...
    Constants) %#codegen

% 1. CALCULATE SOLUTION PROPERTIES IN THE CHANNELS (BULK) AND INTERFACES

% The output of each function below is added to an existing structure as several new fields.

[State_out.Asln] = FUNC_PROPERTIES_BULK(State.Asln, Grid_size, Species.fuel, ...
    Properties.anode, Pointer.anode, Flags, Constants);
[State_out.Csln] = FUNC_PROPERTIES_BULK(State.Csln, Grid_size, Species.oxidizer, ...
    Properties.cathode, Pointer.cathode, Flags, Constants);

% Calculate properties of electrolyte solution at the interfaces.
[State_out.A_int] = FUNC_PROPERTIES_INTERFACES(State.A_int, Grid_size, Species.fuel, ...
    Properties.anode, Pointer.anode, Flags, Constants);
[State_out.M_int_a] = FUNC_PROPERTIES_INTERFACES(State.M_int_a, Grid_size, Species.fuel, ...
    Properties.anode, Pointer.anode, Flags, Constants);
[State_out.M_int_c] = FUNC_PROPERTIES_INTERFACES(State.M_int_c, Grid_size, Species.oxidizer, ...
    Properties.cathode, Pointer.cathode, Flags, Constants);
[State_out.C_int] = FUNC_PROPERTIES_INTERFACES(State.C_int, Grid_size, Species.oxidizer, ...
    Properties.cathode, Pointer.cathode, Flags, Constants);

State_out.C = State.C;

% 2. CALCULATE FLUXES ACROSS SCALAR CELL BOUNDARIES DUE TO MIGRATION, DIFFUSION AND ELECTRODE RXNS

Yfluxes = FUNC_Y_DIFF_MIG_REACT_FLUXES(State_out.Asln, State_out.M_int_a, State_out.M_int_c, ...

```

```

    State_out.Csln, Geometry, Grid_size, Species, Properties, Pointer, Constants, Flags );

Xfluxes = FUNC_X_DIFF_MIG_FLUXES(State_out.Asln, State_out.Csln, Geometry, Grid_size, Species, ...
    Properties, Pointer, Constants, Flags);

% 3. CALCULATE THE MEMBRANE MASS FLUXES

Membrane_mass_flux.am = sum(Yfluxes.mass.am,2)'; % kg/(m^2 s)
Membrane_mass_flux.cm = -sum(Yfluxes.mass.cm,2)'; % kg/(m^2 s)
% NOTE: Negative sign accounts for change in coordinate system

% 4. CALCULATE VALUES AT SCALAR CELL BOUNDARIES

% The mass and species conservation equations ensure those properties are conserved over the scalar
% cells. To evaluate the equations, we need to know the mass density, mass fractions, and total mass
% fluxes at the scalar cell boundaries. The FUNC_SCB_VALUES function calculates the properties by
% linear interpolation between cell centers (accounting for different cell sizes) and the mass fluxes
% are the sum of advection, diffusion and migration mass fluxes at the boundaries. The last levels
% in each structure are .x and .y, which signify whether the values cell boundaries in the
% x-direction or y-direction.

[SCB.asln.mass_flux, SCB.asln.mass_density, SCB.asln.Mass_frac, SCB.asln.charge_density] = ...
    FUNC_SCB_VALUES( State_out.Asln, Xfluxes.mass.asln, Yfluxes.mass.asln, Geometry, Grid_size, ...
    Species.fuel );
[SCB.csln.mass_flux, SCB.csln.mass_density, SCB.csln.Mass_frac, SCB.csln.charge_density] = ...
    FUNC_SCB_VALUES( State_out.Csln, Xfluxes.mass.csln, Yfluxes.mass.csln, Geometry, Grid_size, ...
    Species.oxidizer );

% 5. CALCULATE VALUES AT VELOCITY CELL BOUNDARIES

[VCB_x_asln_x_vel, VCB_y_asln_y_vel] = FUNC_VCB_VALUES( State_out.Asln, Grid_size );
[VCB_x_csln_x_vel, VCB_y_csln_y_vel] = FUNC_VCB_VALUES( State_out.Csln, Grid_size );

VCB.x.asln.x_vel = VCB_x_asln_x_vel;
VCB.x.csln.x_vel = VCB_x_csln_x_vel;
VCB.y.asln.y_vel = VCB_y_asln_y_vel;

```

```

    VCB.y.csln.y_vel = VCB_y_csln_y_vel;
end

function [Sln] = FUNC_PROPERTIES_BULK(Sln, Grid_size, Species, Properties, Pointer, Flags, Constants)

%% SUMMARY: DBFC_PROPERTIES_BULK
% Purpose: This script calculates the electrolyte solution properties, given the state variables in the
% solution vector SV.
% Author: Rick Stroman

%% 1. PRE-ALLOCATE MEMORY FOR MATRICES CREATED IN THIS FUNCTION TO SPEED UP THE CODE

% These bounds are chosen so that the properties matrices have the same size as the state variable
% matrices, to simplify calculations later.
x_d_ub = Grid_size.x_d_num + 2;
y_d_ub = Grid_size.y_d_num + 2;

% Initialize mass density
Sln.mass_density = zeros(x_d_ub, y_d_ub);

% Initialize mole fractions
Sln.Mole_fracs = zeros(x_d_ub, y_d_ub, Species.num);

% Initialize mole densities (concentrations)
Sln.Mole_densities = zeros(x_d_ub, y_d_ub, Species.num);

% Initialize ionic strength
Sln.ionic_strength = zeros(x_d_ub, y_d_ub);

% Initialize activity coefficients (note the are initialized to an ideal solution... all ones)
Sln.Act_coeffs = ones(x_d_ub, y_d_ub, Species.num);

% Initialize the net charge density of solution

```

```

Sln.charge_density = zeros(x_d_ub, y_d_ub);

% Use these column vectors to change the dimensions of Mass_fracs and Mole_densities inside the loop
% instead of using squeeze, which is very expensive.
Mass_fracs      = zeros(Species.num, 1);
Mole_densities = zeros(Species.num, 1);

%% 2. CALCULATE ELECTROLYTE PROPERTIES

for x_d = 1 : Grid_size.x_d_num + 2
    % Include the inlet ghost cells, which have been assigned BC mass fractions, but ignore the outlet
    % ghost cells, in which solution properties are never needed.

    for y_d = 1 : Grid_size.y_d_num + 2
        % Include the anode and cathode interfaces and both membrane interfaces.
        Mass_fracs(:) = Sln.Mass_fracs(x_d,y_d,:);

        % DENSITY AND CONCENTRATION RELATED PROPERTIES

        % Mass density - [kg/m^3]
        Sln.mass_density(x_d,y_d) = FUNC_SOLUTION_MASS_DENSITY(Mass_fracs, Properties, Pointer);

        % Species mole fractions - [unitless]
        Sln.Mole_fracs(x_d,y_d,:) = FUNC_MASS_TO_MOLE_FRACTIONS(Mass_fracs, Properties.Molar_mass, ...
            Species);

        % Species mole densities - [kmol/m^3]
        Sln.Mole_densities(x_d,y_d,:) = FUNC_SPECIES_MOLE_DENSITIES(Sln.mass_density(x_d,y_d), ...
            Mass_fracs, Properties.Molar_mass);
        Mole_densities(:) = Sln.Mole_densities(x_d, y_d, :); % Making it a vector for local use this
                                                                % way because squeeze is very expensive

        % ELECTROLYTE RELATED PROPERTIES

        if ~Flags.model.solution_ideality

```

```

% Ionic strength - [kmol/m^3]
Sln.ionic_strength(x_d,y_d) = 0.5 * Mole_densities' * Properties.Electric_charge.^2;

% Activity coefficients - [unitless]
% Sln.Act_coeffs(x_d,y_d,:) = FUNC_DEBYE_HUCKEL(Sln.ionic_strength(x_d,y_d), Properties, ...
% Constants);
Sln.Act_coeffs(x_d,y_d,:) = FUNC_H2O2_Act_Coeffs(Species, Sln.Mole_fracs(x_d, y_d, :), ...
    Constants);

end % If we don't model the non-ideality of the solution, the activity coefficients keep ...
    % their initialized values... 1.

% Calculate the net charge density in each phase for each x-discretization - [C/m^3]
Sln.charge_density(x_d,y_d) = Constants.faraday * Properties.Electric_charge_T * Mole_densities;

end

end

end

function [Int] = FUNC_PROPERTIES_INTERFACES(Int, Grid_size, Species, Properties, Pointer, Flags, ...
    Constants) %#codegen

%% SUMMARY: DBFC_PROPERTIES_INTERFACES
% Purpose: This script calculates the electrolyte solution properties at the electrode and membrane
% interfaces, from the mass fractions and/or mole fractions.
% Author: Rick Stroman

% Initialize mass density
Int.mass_density = zeros(1,Grid_size.x_d_num);

% Initialize mole fractions
Int.Mole_fracs = zeros(Grid_size.x_d_num, Species.num);

```

```

% Initialize mole densities (concentrations)
Int.Mole_densities = zeros(Grid_size.x_d_num, Species.num);

% Initialize ionic strength
Int.ionic_strength = zeros(1,Grid_size.x_d_num);

% Initialize activity coefficients (note the are initialized to an ideal solution... all ones)
Int.Act_coeffs = ones(Grid_size.x_d_num, Species.num);
% Initialize the net charge density of solution
Int.charge_density = zeros(1,Grid_size.x_d_num);

%% 1. CALCULATE PROPERTIES AT INTERFACES

Mass_fracs = Int.Mass_fracs';

for x_d = 1:Grid_size.x_d_num

    % Species mole fractions - [unitless]
    Int.Mole_fracs(x_d,:) = FUNC_MASS_TO_MOLE_FRACTIONS(Mass_fracs(:,x_d), Properties.Molar_mass, ...
        Species);

    % Mass density - [kg/m^3]
    Int.mass_density(x_d) = FUNC_SOLUTION_MASS_DENSITY(Mass_fracs(:,x_d), Properties, Pointer);

    % Species mole densities - [unitless]
    Int.Mole_densities(x_d,:) = FUNC_SPECIES_MOLE_DENSITIES(Int.mass_density(x_d), Mass_fracs(:,x_d), ...
        Properties.Molar_mass);

    if ~Flags.model.solution_ideality

        % Ionic strength - [kmol/m^3]
        Int.ionic_strength(x_d) = 0.5 * Int.Mole_densities(x_d,:) * Properties.Electric_charge.^2;

        % Activity coefficients - [unitless]
        %Int.Act_coeffs(x_d,:) = FUNC_DEBYE_HUCKEL(Int.ionic_strength(x_d), Properties, Constants);

```

```

    Int.Act_coeffs(x_d,:) = FUNC_H2O2_Act_Coeffs(Species, Int.Mole_fracs(x_d,:), Constants);

end

% Calculate the net charge density in each phase for each x-discretization - [C/m^3]
Int.charge_density(x_d) = Constants.faraday * Int.Mole_densities(x_d,:) * Properties.Electric_charge;

end
end

function Yfluxes = FUNC_Y_DIFF_MIG_REACT_FLUXES(Asln, M_int_a, M_int_c, Csln, ...
        Geometry, Grid_size, Species, Properties, Pointer, Constants, Flags) %#codegen

%% SUMMARY: FUNC_DIFF_MIG_REACT_FLUXES
% Purpose: This script calculates the y-direction fluxes due to diffusion, migration and (at the
% electrode surfaces) reactions.
% Author: Rick Stroman
% Date: 28 October 2011

%% NOTES
%
% 1. Indexing scheme: Same as for velocity in the electrolyte solution, where each flux is indexed by
% x_discretization, y_discretization and then species. Fluxes are assumed to be at the same locations as
% the y-velocity in the electrolyte solution. Fluxes with the index .asln(:,1,:) and
% .csln(:,Grid_size.y_d_num+2,:) are ghost cells for the electrolyte solution, but here are treated as
% the fluxes from the electrode interfaces to the bulk solution. The same goes for
% .asln(:,Grid_size.y_d_num+2,:) and .csln(:,1,:), which are fluxes into and out of the membrane
% interfaces, respectively.
%
% 2. Flux structures with .asln and .csln fields are in the electrolyte solution or interfaces. Flux
% structures with fields .arxn and .crxn are fluxes to/from the electrodes due to reactions at the
% surfaces. The field .m refers to the flux through the membrane. The fields .arxn, .crxn and .m have
% no y-index.
%
```

```
% 3. I tried to vectorize the loops, but MATLAB wouldn't let me put the results from the flux calcs back
% into the flux matrices... whenever a flux matrix has a singleton dimension, it ignores it, even if it
% changes from one operation to the next... so I would have to calculate ALL of the fluxes at once and
% stuff them back into the matrix. This is probably possible, but will take some more thought.
```

```
%% 1. SET RANGES AND PRE-ALLOCATE MEMORY FOR VECTORS CHANGED IN LOOPS TO SPEED UP THE CODE
```

```
% Initialize mass fluxes
```

```
Yfluxes.mass.asln = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.fuel.num);
Yfluxes.mass.csln = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.oxidizer.num);
Yfluxes.mass.arxn = zeros(Grid_size.x_d_num, Species.fuel.num);
Yfluxes.mass.crxn = zeros(Grid_size.x_d_num, Species.oxidizer.num);
Yfluxes.mass.am   = zeros(Grid_size.x_d_num, Species.fuel.num);
Yfluxes.mass.cm   = zeros(Grid_size.x_d_num, Species.oxidizer.num);
Yfluxes.mass_diff.asln = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.fuel.num);
Yfluxes.mass_diff.csln = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.oxidizer.num);
Yfluxes.mass_mig.asln  = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.fuel.num);
Yfluxes.mass_mig.csln  = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.oxidizer.num);
```

```
% Initialize mole fluxes
```

```
Yfluxes.mole.asln = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.fuel.num);
Yfluxes.mole.csln = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.oxidizer.num);
Yfluxes.mole.arxn = zeros(Grid_size.x_d_num, Species.fuel.num);
Yfluxes.mole.crxn = zeros(Grid_size.x_d_num, Species.oxidizer.num);
Yfluxes.mole.am   = zeros(Grid_size.x_d_num, Species.fuel.num);
Yfluxes.mole.cm   = zeros(Grid_size.x_d_num, Species.oxidizer.num);
Yfluxes.mole.mem_mig_Na = zeros(Grid_size.x_d_num, 1);
Yfluxes.mole.mem_diff_Na = zeros(Grid_size.x_d_num, 1);
Yfluxes.mole_diff.asln = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.fuel.num);
Yfluxes.mole_diff.csln = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.oxidizer.num);
Yfluxes.mole_mig.asln  = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.fuel.num);
Yfluxes.mole_mig.csln  = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.oxidizer.num);
```

```
Mole_fluxes_to_interface_a = zeros(Grid_size.x_d_num, Species.fuel.num);
Mole_fluxes_to_interface_c = zeros(Grid_size.x_d_num, Species.oxidizer.num);
```

```

% Initialize charge fluxes
Yfluxes.charge.asln = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1);
Yfluxes.charge.csln = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1);
Yfluxes.charge.arxn = zeros(Grid_size.x_d_num, 1);
Yfluxes.charge.crxn = zeros(Grid_size.x_d_num, 1);
Yfluxes.charge.m     = zeros(Grid_size.x_d_num, 1);
Yfluxes.charge.am    = zeros(Grid_size.x_d_num, 1);
Yfluxes.charge.cm    = zeros(Grid_size.x_d_num, 1);

% Create column vectors used locally to avoid having to call squeeze, which is expensive.
Act_coeffs_a_1 = ones(Species.fuel.num, 1);
Act_coeffs_a_2 = ones(Species.fuel.num, 1);
Mole_densities_a_1 = ones(Species.fuel.num, 1);
Mole_densities_a_2 = ones(Species.fuel.num, 1);

Act_coeffs_c_1 = ones(Species.oxidizer.num, 1);
Act_coeffs_c_2 = ones(Species.oxidizer.num, 1);
Mole_densities_c_1 = ones(Species.oxidizer.num, 1);
Mole_densities_c_2 = ones(Species.oxidizer.num, 1);

Yfluxes_mass_asln = ones(Species.fuel.num, 1);
Yfluxes_mass_csln = ones(Species.oxidizer.num, 1);
Yfluxes_mole_asln = ones(Species.fuel.num, 1);
Yfluxes_mole_csln = ones(Species.oxidizer.num, 1);

anode_species_range = 1 : Species.fuel.num;
cathode_species_range = 1 : Species.oxidizer.num;
mem_species_range = 1 : Species.membrane.num;

%% 2. MASS FLUXES IN THE BULK ELECTROLYTE SOLUTION [kg/(m^2 s)]

for x_d = 2 : Grid_size.x_d_num + 1 % Step through indexes of real fluid cells in the x-direction.

    for y_d = 1 : Grid_size.y_d_num + 1

        % Step through the bulk in the direction from anode to cathode, one y-discretization at a time. Note

```

```

% that we need the fluxes into and out of each cell, so the total number of fluxes is the number of
% cells plus 1. Each flux is assumed to be entering the cell of the same index... i.e. flux y_d = 3
% is entering cell y_d = 3. This indexing scheme was chosen because I can't have an index of 0.

% y_d = 1 is the flux out of the electrode-solution interface and into the first fluid cell ( cell
% (:,1) ). It is calculated using properties at the interface and in the first fluid cell. This is
% the flux which the solver equates to the reaction flux to solve for mass fractions at the
% electrode-solution interface.

% y_d = 2 is the flux out of the first fluid cell nearest the electrode.

% y_d = Grid_size.y_d_num is the flux into the fluid cell nearest the membrane.

% y_d = Grid_size.y_d_num + 1 is the flux out of the fluid cell nearest the membrane, and into the
% solution-membrane interface. It is calculated using properties in the last fluid cell and at the
% solution-membrane interface. The solver will find the species mass fractions at the interface by
% equating this flux with the membrane flux.

% ANODE SIDE

% Copy the relevent portions of 3D matrices into vectors to avoid using the squeeze function
Act_coeffs_a_1(anode_species_range,1) = Asln.Act_coeffs(x_d,y_d, anode_species_range);
Act_coeffs_a_2(anode_species_range,1) = Asln.Act_coeffs(x_d,y_d + 1,anode_species_range);
Mole_densities_a_1(anode_species_range,1) = Asln.Mole_densities(x_d,y_d, anode_species_range);
Mole_densities_a_2(anode_species_range,1) = Asln.Mole_densities(x_d,y_d + 1,anode_species_range);

% Calculate the anode side mass fluxes in the y-direction due to diffusion and migration
[ Yfluxes.mass.asln(x_d-1,y_d,anode_species_range), ...
  Yfluxes.mass_diff.asln(x_d-1,y_d,anode_species_range), ...
  Yfluxes.mass_mig.asln(x_d-1,y_d,anode_species_range), ...
  Yfluxes.mole_diff.asln(x_d-1,y_d,anode_species_range), ...
  Yfluxes.mole_mig.asln(x_d-1,y_d,anode_species_range) ] = FUNC_ELECTROLYTE_MASS_FLUXES( ...
    Act_coeffs_a_1, ...
    Act_coeffs_a_2, ...
    Mole_densities_a_1, ...
    Mole_densities_a_2, ...

```

```

Asln.elec_pot(x_d,y_d ), ...
Asln.elec_pot(x_d,y_d+1), ...
Geometry.y_d_size(y_d ), ...
Geometry.y_d_size(y_d+1), ...
Properties.anode, ...
Pointer.anode, ...
Constants, ...
Species.fuel, ...
Flags.model.y);

% CATHODE SIDE

% Copy the relevent portions of 3D matrices into vectors to avoid using the squeeze function
Act_coeffs_c_1(cathode_species_range,1) = Csln.Act_coeffs(x_d,y_d, cathode_species_range,1);
Act_coeffs_c_2(cathode_species_range,1) = Csln.Act_coeffs(x_d,y_d + 1,cathode_species_range,1);
Mole_densities_c_1(cathode_species_range,1) = Csln.Mole_densities(x_d,y_d, ...
                                                                cathode_species_range,1);
Mole_densities_c_2(cathode_species_range,1) = Csln.Mole_densities(x_d,y_d + 1,...
                                                                cathode_species_range,1);

% Calculate the cathode side mass fluxes in the y-direction due to diffusion and migration
[ Yfluxes.mass.csln(x_d-1,y_d,cathode_species_range,1), ...
  Yfluxes.mass_diff.csln(x_d-1,y_d,cathode_species_range,1), ...
  Yfluxes.mass_mig.csln(x_d-1,y_d,cathode_species_range,1), ...
  Yfluxes.mole_diff.csln(x_d-1,y_d,cathode_species_range,1), ...
  Yfluxes.mole_mig.csln(x_d-1,y_d,cathode_species_range,1) ] = FUNC_ELECTROLYTE_MASS_FLUXES( ...
  Act_coeffs_c_1, ...
  Act_coeffs_c_2, ...
  Mole_densities_c_1, ...
  Mole_densities_c_2, ...
  Csln.elec_pot(x_d, y_d), ...
  Csln.elec_pot(x_d, y_d+1), ...
  Geometry.y_d_size(y_d ), ...
  Geometry.y_d_size(y_d+1), ...
  Properties.cathode, ...
  Pointer.cathode, ...

```

```

        Constants,          ...
        Species.oxidizer,  ...
        Flags.model.y);
end

end

%% 3. COMPUTE MOLE AND CHARGE FLUXES IN THE BULK FROM THE MASS FLUXES IN THE BULK

for x_d = 1 : Grid_size.x_d_num

    for y_d = 1 : Grid_size.y_d_num + 1

        % ELECTROLYTE SOLUTION AND INTERFACES

        % Copy the species mass fluxes into local variables to avoid using the squeeze function
        Yfluxes_mass_asln(anode_species_range,1) = Yfluxes.mass.asln(x_d,y_d,anode_species_range);
        Yfluxes_mass_csln(cathode_species_range,1) = Yfluxes.mass.csln(x_d,y_d,cathode_species_range);

        % Compute the electrolyte mole fluxes [kmol/(m^2 s)]
        Yfluxes.mole.asln(x_d,y_d,anode_species_range) = Yfluxes_mass_asln ...
            .* Properties.anode.one_over_molar_mass(anode_species_range);
        Yfluxes.mole.csln(x_d,y_d,cathode_species_range) = Yfluxes_mass_csln ...
            .* Properties.cathode.one_over_molar_mass(cathode_species_range);

        % Copy the species mole fluxes into local variables to avoid using the squeeze function
        Yfluxes_mole_asln(anode_species_range,1) = Yfluxes.mole.asln(x_d,y_d,anode_species_range);
        Yfluxes_mole_csln(cathode_species_range,1) = Yfluxes.mole.csln(x_d,y_d,cathode_species_range);

        % Compute the electrolyte charge fluxes [C/(m^2 s)]
        Yfluxes.charge.asln(x_d,y_d) = Constants.faraday * Properties.anode.Electric_charge_T * ...
            Yfluxes_mole_asln;
        Yfluxes.charge.csln(x_d,y_d) = Constants.faraday * Properties.cathode.Electric_charge_T * ...
            Yfluxes_mole_csln;

    end

end

```

```
end
```

```
%% 4. MASS FLUXES THROUGH THE MEMBRANE [kg/(m^2 s)]
```

```
% Note that this function calculates the mass fluxes for the whole length of the channel at once.
```

```
% The matrices returned from this function have the indices (x_d, species).
```

```
[ Yfluxes.mass.am, Yfluxes.mass.cm, Yfluxes.mole.mem_mig_Na, Yfluxes.mole.mem_diff_Na ] = ...  
  FUNC_MEMBRANE_MASS_FLUXES( ... % Membrane flux with anode and cathode species orders  
  M_int_a           , M_int_c,           ...  
  Geometry.mem_thick , Properties,       ...  
  Constants          , Species,          ...  
  Pointer            , Flags,            ...  
  Grid_size);
```

```
%% 5. COMPUTE MOLE AND CHARGE FLUXES AT THE MEMBRANE AND ELECTRODES FROM THE MASS FLUXES
```

```
for x_d = 1 : Grid_size.x_d_num
```

```
  % REACTIONS AND MEMBRANE - no y-index dependence
```

```
  % Compute membrane mole fluxes [kmol/(m^2 s)]
```

```
  Yfluxes.mole.am(x_d,anode_species_range) = ...  
    Yfluxes.mass.am(x_d,1:length(Properties.anode.one_over_molar_mass)) ...  
    .* Properties.anode.one_over_molar_mass';
```

```
  Yfluxes.mole.cm(x_d,cathode_species_range) = ...  
    Yfluxes.mass.cm(x_d,1:length(Properties.cathode.one_over_molar_mass)) ...  
    .* Properties.cathode.one_over_molar_mass';
```

```
  % Compute the membrane charge flux [C/(m^2 s)]
```

```
  Yfluxes.charge.am(x_d,1) = ...  
    Constants.faraday * Properties.anode.Electric_charge' * Yfluxes.mole.am(x_d,:);  
  Yfluxes.charge.cm(x_d,1) = ...  
    Constants.faraday * Properties.cathode.Electric_charge' * Yfluxes.mole.cm(x_d,:);
```

```
End
```

```

function Xfluxes = FUNC_X_DIFF_MIG_FLUXES(Asln, Csln, Geometry, Grid_size, Species, ...
                                         Properties, Pointer, Constants, Flags) %#codegen

%% SUMMARY: FUNC_X_DIFF_MIG_FLUXES
% Purpose: This script calculates the x-direction fluxes due to diffusion and migration.
% Author: Rick Stroman

%% NOTES
%
% 1. Indexing scheme: Each flux is indexed by x_discretization, y_discretization and then species.
% x-direction fluxes are assumed to be at the same locations as the x-velocity in the electrolyte
% solution, namely at the boundaries between x-discretizations.
%
% 2. Some discretizations are ghost cells, which store the values at interfaces. They are used here
% to calculate the fluxes to or from the interfaces on the bulk electrolyte side of the interface.
% .asln(:,1,:) --- anode interface
% .asln(:,Grid_size.y_d_num+2,:) --- anode side membrane interface
% .asln(1,,:) --- anode inlet
% .asln(Grid_size.x_d_num+2,,:) --- anode outlet
% .csln(:,1,:) --- cathode side membrane interface
% .csln(:,Grid_size.y_d_num+2,:) --- cathode interface
% .csln(1,,:) --- cathode inlet
% .csln(Grid_size.x_d_num+2,,:) --- cathode outlet
%
% 3. Flux directions: x-fluxes are positive in the direction from inlet to outlet, and y-fluxes are
% positive in the direction from anode to cathode.
%
% 4. Flux structures with .asln and .csln fields are in the electrolyte solution. Flux structures with
% fields .arxn and .crxn are fluxes to/from the electrodes due to reactions at the surfaces. The field
% .m refers to the flux through the membrane. The fields .arxn, .crxn and .m have no y-index, because
% they are linear in the x-direction.
%
% 5. I tried to vectorize the loops, but MATLAB wouldn't let me put the results from the flux calcs back
% into the flux matrices... whenever a flux matrix has a singleton dimension, it ignores it, even if it
% changes from one operation to the next... so I would have to calculate ALL of the fluxes at once and
% stuff them back into the matrix. This is probably possible, but will take some more thought.

```

```

%% 1. SET RANGES AND PRE-ALLOCATE MEMORY FOR VECTORS CHANGED IN LOOPS TO SPEED UP THE CODE

% Initialize mass fluxes
Xfluxes.mass.asln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.fuel.num);
Xfluxes.mass.csln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.oxidizer.num);
Xfluxes.mass_diff.asln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.fuel.num);
Xfluxes.mass_diff.csln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.oxidizer.num);
Xfluxes.mass_mig.asln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.fuel.num);
Xfluxes.mass_mig.csln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.oxidizer.num);

% Initialize mole fluxes
Xfluxes.mole.asln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.fuel.num);
Xfluxes.mole.csln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.oxidizer.num);
Xfluxes.mole_diff.asln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.fuel.num);
Xfluxes.mole_diff.csln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.oxidizer.num);
Xfluxes.mole_mig.asln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.fuel.num);
Xfluxes.mole_mig.csln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.oxidizer.num);

% Initialize charge fluxes
Xfluxes.charge.asln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num);
Xfluxes.charge.csln = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num);

% Create column vectors used locally to avoid having to call squeeze, which is expensive.
Act_coeffs_a_1 = ones(Species.fuel.num, 1);
Act_coeffs_a_2 = ones(Species.fuel.num, 1);
Mole_densities_a_1 = ones(Species.fuel.num, 1);
Mole_densities_a_2 = ones(Species.fuel.num, 1);

Act_coeffs_c_1 = ones(Species.oxidizer.num, 1);
Act_coeffs_c_2 = ones(Species.oxidizer.num, 1);
Mole_densities_c_1 = ones(Species.oxidizer.num, 1);
Mole_densities_c_2 = ones(Species.oxidizer.num, 1);

Xfluxes_mass_asln = ones(Species.fuel.num, 1);

```

```

Xfluxes_mass_csln = ones(Species.oxidizer.num, 1);
Xfluxes_mole_asln = ones(Species.fuel.num, 1);
Xfluxes_mole_csln = ones(Species.oxidizer.num, 1);

%% 2. MASS FLUXES IN THE BULK ELECTROLYTE SOLUTION [kg/(m^2 s)]

% Step down the channel from the inlet to the outlet, one x-discretization at a time. The index x_d
% steps through the indices of the real discretizations (not including the ghost cells at each end). Note
% that we need the fluxes into and out of each cell, so the total number of fluxes is the number of cells
% plus 1. Each flux is assumed to be entering the cell of the same index... i.e. flux x_d = 3 is
% entering cell x_d = 3. This indexing scheme was chosen because I can't have an index of 0.

% x_d = 1 is the flux from the inlet into the first fluid cell. It is calculated using properties
% at the inlet and in the first fluid cell.
% x_d = 2 is the flux out of the first fluid cell nearest the inlet.
% x_d = Grid_size.x_d_num is the flux into the last fluid cell near the outlet.
% x_d = Grid_size.x_d_num + 1 is the flux out of the fluid cell nearest the outlet. It is calculated
% using properties in the last fluid cell.

for x_d = 2 : Grid_size.x_d_num + 1 % Step through the x-direction fluxes. Neglect x_d = 1 to eliminate
    % unrealistic fluxes at inlet which don't obey charge neutrality
    % upstream of the model domain.

    for y_d = 2 : Grid_size.y_d_num + 1 % Step through the y-direction indexes of real fluid cells

        % Mass fluxes of each species

        Act_coeffs_a_1(:) = Asln.Act_coeffs(x_d,y_d,:);
        Act_coeffs_a_2(:) = Asln.Act_coeffs(x_d+1,y_d,:);
        Mole_densities_a_1(:) = Asln.Mole_densities(x_d,y_d,:);
        Mole_densities_a_2(:) = Asln.Mole_densities(x_d+1,y_d,:);

        [Xfluxes.mass.asln(x_d,y_d-1,:), ...
        Xfluxes.mass_diff.asln(x_d,y_d-1,:), ...

```

```

Xfluxes.mass_mig.asln(x_d,y_d-1,:), ...
Xfluxes.mole_diff.asln(x_d,y_d-1,:), ...
Xfluxes.mole_mig.asln(x_d,y_d-1,:) ] = FUNC_ELECTROLYTE_MASS_FLUXES( ...
    Act_coeffs_a_1, ...
    Act_coeffs_a_2, ...
    Mole_densities_a_1, ...
    Mole_densities_a_2, ...
    Asln.elec_pot(x_d,y_d), ...
    Asln.elec_pot(x_d+1,y_d), ...
    Geometry.x_d_size(x_d), ...
    Geometry.x_d_size(x_d+1), ...
    Properties.anode, ...
    Pointer.anode, ...
    Constants, ...
    Species.fuel, ...
    Flags.model.x);

```

% Mass fluxes of each species

```

Act_coeffs_c_1(:) = Csln.Act_coeffs(x_d,y_d,:);
Act_coeffs_c_2(:) = Csln.Act_coeffs(x_d+1, y_d,:);
Mole_densities_c_1(:) = Csln.Mole_densities(x_d,y_d,:);
Mole_densities_c_2(:) = Csln.Mole_densities(x_d+1, y_d,:);

[Xfluxes.mass.csln(x_d,y_d-1,:), ...
Xfluxes.mass_diff.csln(x_d,y_d-1,:), ...
Xfluxes.mass_mig.csln(x_d,y_d-1,:), ...
Xfluxes.mole_diff.csln(x_d,y_d-1,:), ...
Xfluxes.mole_mig.csln(x_d,y_d-1,:) ] = FUNC_ELECTROLYTE_MASS_FLUXES( ...
    Act_coeffs_c_1, ...
    Act_coeffs_c_2, ...
    Mole_densities_c_1, ...
    Mole_densities_c_2, ...
    Csln.elec_pot(x_d,y_d), ...

```

```

                                Csln.elec_pot(x_d+1, y_d),      ...
                                Geometry.x_d_size(x_d),        ...
                                Geometry.x_d_size(x_d+1),      ...
                                Properties.cathode,             ...
                                Pointer.cathode,               ...
                                Constants,                     ...
                                Species.oxidizer,              ...
                                Flags.model.x);
end
end

%% 3. COMPUTE MOLE AND CHARGE FLUXES FROM THE MASS FLUXES

for x_d = 1 : Grid_size.x_d_num + 1

    for y_d = 1 : Grid_size.y_d_num

        % ELECTROLYTE SOLUTION AND INTERFACES

        Xfluxes_mass_asln(:) = Xfluxes.mass.asln(x_d,y_d,:);
        Xfluxes_mass_csln(:) = Xfluxes.mass.csln(x_d,y_d,:);

        % Compute the electrolyte mole fluxes [kmol/(m^2 s)]
        Xfluxes.mole.asln(x_d,y_d,:) = ...
            Xfluxes_mass_asln(length(Properties.anode.one_over_molar_mass)) ...
            .* Properties.anode.one_over_molar_mass;
        Xfluxes.mole.csln(x_d,y_d,:) = ...
            Xfluxes_mass_csln(length(Properties.cathode.one_over_molar_mass)) ...
            .* Properties.cathode.one_over_molar_mass;

        Xfluxes_mole_asln(:) = Xfluxes.mole.asln(x_d,y_d,:);
        Xfluxes_mole_csln(:) = Xfluxes.mole.csln(x_d,y_d,:);
    end
end

```

```

    % Compute the electrolyte charge fluxes [C/(m^2 s)]
    Xfluxes.charge.asln(x_d,y_d) = Constants.faraday * Properties.anode.Electric_charge_T ...
    * Xfluxes_mole_asln;
    Xfluxes.charge.csln(x_d,y_d) = Constants.faraday * Properties.cathode.Electric_charge_T ...
    * Xfluxes_mole_csln;

end

end

function [Mass_fluxes, Mass_fluxes_diffusion, Mass_fluxes_migration, Mole_fluxes_diffusion, ...
Mole_fluxes_migration] = FUNC_ELECTROLYTE_MASS_FLUXES( ...
    Act_coeffs_1 , Act_coeffs_2, Mole_densities_1, Mole_densities_2, Elec_pot_1, Elec_pot_2, ...
    cell_y_size_1, cell_y_size_2, Properties , Pointer, Constants , Species, Flags) %#codegen

```



```

% | | |
% -----
%
% (5) The squeeze function applied to the electrolyte solution properties produces a matrix which is
% indexed (x_d, s_i), so that is the format of Act_coeffs and Mole_densities

%% 1. CALCULATE LOCALLY USEFUL QUANTITIES

Activities_1 = zeros(Species.num,1);
Activities_2 = zeros(Species.num,1);

% Calculate activities of all species in (:,y_1,:) and (:,y_2,:)
Activities_1 = Act_coeffs_1 .* Mole_densities_1;
Activities_2 = Act_coeffs_2 .* Mole_densities_2;

% Calculate the distance between the centers of the two cells defining the flux across thier boundary.
one_over_delta_y = 2 / (cell_y_size_1 + cell_y_size_2);

% Calculate the mole density of each species at the boundary between cells, assuming a linear gradient.
% Same process as above.
Boundary_Mole_densities = Mole_densities_2 - ...
    0.5 * cell_y_size_2 * (Mole_densities_2 - Mole_densities_1) * one_over_delta_y;

%% 2. CALCULATE THE DIFFUSION FLUX OF EACH SPECIES FROM Y-DISCRETIZATION "1" TO Y-DISCRETIZATION "2"

% Molar flux due to diffusion in responseto the activity gradient (this is a vector). See Ref [1] pg
% 29. Sign check: When Activities_1 > Acitivities_2, then the flux should be positive. Example: when the
% activity in asln_a > asln, then the flow should be from asln_a to asln.
Mole_fluxes_diffusion = Flags.diffusion * Properties.Diffusivities .* -(Activities_2 - Activities_1) ...
    * one_over_delta_y; % kmol/(m^2 s)

%Mole_fluxes_diffusion = Flags.diffusion * Properties.Diffusivities .* -(Mole_densities_2 -
Mole_densities_1) * one_over_delta_y; % kmol/(m^2 s)

% The water flux cannot be calculated accurately using Fick's law because the concentration is enormous

```

```

% compared to everything else, and changes slightly from one cell to the next. Zero out the erroneous
% mole flux of water. Calculate the mass flux of water in step #4 and stick it into the mass flux vector
% then.
Mole_fluxes_diffusion(Pointer.species.H2O,1) = 0;

% Calculate the mass fluxes of non-water species
Mass_fluxes_diffusion = Mole_fluxes_diffusion(1:Species.num,1) .* Properties.Molar_mass; % kg/(m^2 s)

% The net diffusion mass flux must be zero to conserve momentum, so we use water to balance the mass
% fluxes of everything else. Solutes go one way, and water goes the other way to keep the momentum
% fluxes balanced.

% We want sum(Y_k * rho * v_k) = 0 and mass_flux_diffusion_H2O = sum(mass_flux_diffusion_k) where
% k ~= 0

% Replace the erroneous water diffusion mass flux with the opposite of the total solute mass flux
Mass_fluxes_diffusion(Pointer.species.H2O,1) = - Properties.Molar_mass_T * Mole_fluxes_diffusion;

% Use the water mass flux to find the water mole flux
Mole_fluxes_diffusion(Pointer.species.H2O,1) = Mass_fluxes_diffusion(Pointer.species.H2O,1) ...
    / Properties.Molar_mass(Pointer.species.H2O);

%% 3. CALCULATE THE MIGRATION FLUX OF EACH SPECIES FROM Y-DISCRETIZATION "1" TO Y-DISCRETIZATION "2"

% Molar fluxes due to migration in response to the electric potential gradient (this is a vector). See
% Ref [1] pg 29. Note that z_i * F / (R * T) gives the mobility due to the Nernst - Einstein relation,
% and the mobility is the terminal velocity of an ion in solution in response to a force of 1 N... or
% alternatively, in response to a an electric field (potential gradient) of 1 V. See Ref [1] pg 66 and
% Ref [2] pg 11 and 283. The complete equation for migration is discussed in Ref [2] chapter 11
% "Infinitely dilute solutions".
Mole_fluxes_migration = Properties.Diffusivities * Constants.FoRT ...
    .* Properties.Electric_charge .* Boundary_Mole_densities ...
    * -(Elec_pot_2 - Elec_pot_1) * one_over_delta_y;

```

```

% Calculate the mass fluxes due to migration
Mass_fluxes_migration = Mole_fluxes_migration .* Properties.Molar_mass; % kg/(m^2 s)

%% 4. CALCULATE THE TOTAL FLUX OF EACH SPECIES FROM Y-DISCRETIZATION "1" TO Y-DISCRETIZATION "2"

Mass_fluxes = Mass_fluxes_diffusion + Mass_fluxes_migration; % kg/(m^2 s)

end

function [Mass_fluxes_a, Mass_fluxes_c, mig_mole_flux_Na, dif_mole_flux_Na] = ...
    FUNC_MEMBRANE_MASS_FLUXES( M_int_a , M_int_c, thickness_m, Properties,...
        Constants, Species, Pointer, Flags, Grid_size) %#codegen

%% SUMMARY: FUNC_MEMBRANE_MASS_FLUXES
% Purpose: Compute the molar fluxes of each species through the membrane.
% Author: Rick Stroman

%% NOTES
%
% (1) Mass fluxes are returned in units kg/(m^2 s) assuming the inputs to the function are:
%     temperature in K, molar density in kmol/m^3, electric potential in V, binary diffusivity in m^2/s
%     and molar transport coefficient in m/s. Mole fractions and activity coefficients have no units.
%
% (2) At present the only species considered are Na+ and H2O. The rate of Na+ transport is
%     calculated from the membrane conductivity and potential gradient. The rate of H2O transport is
%     calculated by assuming all of the H2O flux is due to electroosmotic drag.
%
% (3) Positive fluxes are directed from the anode to the cathode, and the positive direction is from
%     anode to cathode.

%% 1. CALCULATE THE DIFFUSION AND MIGRATION MASS FLUXES THROUGH THE MEMBRANE

```

```

% Set the cation mole fractions... if we allow H+, these will be solved for and become inputs to the
% function.
M.Mole_fracs.Na = 1;
M.Mole_fracs.H = 0;

% Calculate the cation mobilities using equations from [3]. Note that they have F built-in, so there is
% no need to include it in the migration and diffusion equations!
u_Na = Properties.membrane.Na_mobility * (1 - Properties.membrane.k * M.Mole_fracs.H );
u_H = Properties.membrane.H_mobility * (1 - Properties.membrane.k * M.Mole_fracs.Na);

% Calculate the electric potential and concentration gradients.
elec_grad = (M_int_c.elec_pot - M_int_a.elec_pot) / thickness_m;
conc_grad = (M_int_c.Mole_densities(:,Species.membrane.loc_cathode) - ...
             M_int_a.Mole_densities(:,Species.membrane.loc_anode));

% Migration fluxes
mig_mole_flux_Na = ...
    -Properties.electric_charge.Na * u_Na * M.Mole_fracs.Na * Properties.membrane.SO3_density * elec_grad;
mig_mole_flux_H = ...
    -Properties.electric_charge.H * u_H * M.Mole_fracs.H * Properties.membrane.SO3_density * elec_grad;

% Diffusion fluxes
dif_mole_flux_Na = -u_Na / Constants.faraday * Constants.ideal_gas * Constants.temperature ...
    * conc_grad(:,Pointer.membrane.species.Na);
dif_mole_flux_H2O = -Properties.membrane.H2O_diffusivity * conc_grad(:,Pointer.membrane.species.H2O);
%dif_mole_flux_H = -u_H / Constants.faraday * Constants.ideal_gas * Constants.temperature * ...
% conc_grad(:,Pointer.membrane.species.H);
dif_mole_flux_H = 0; % kludge, because H+ isn't in the list of membrane species, so there is
                    % no pointer for it.

% Total mass fluxes of ions
mass_flux_Na = ( Flags.model.m.migration * mig_mole_flux_Na + ...
    Flags.model.m.diffusion * dif_mole_flux_Na ) * Properties.molar_mass.Na;
mass_flux_H = ( Flags.model.m.migration * mig_mole_flux_H + ...

```

```

Flags.model.m.diffusion * dif_mole_flux_H ) * Properties.molar_mass.H;

%% 2. CALCULATE THE ELECTRO-OSMOTIC DRAG MASS FLUX OF H2O THROUGH THE MEMBRANE

% Electro-osmotic drag flux of H2O
EOD_mole_flux_H2O = mig_mole_flux_Na * Properties.membrane.electro_drag;

%% 3. CALCULATE THE PERMEATION MASS FLUX OF H2O THROUGH THE MEMBRANE

if mean(M_int_a.press) > mean(M_int_c.press)
    water_mass_density = M_int_a.mass_density * M_int_a.Mass_fracs(:,Pointer.anode.species.H2O);
else
    water_mass_density = M_int_c.mass_density * M_int_c.Mass_fracs(:,Pointer.cathode.species.H2O);
end

permeation_mass_flux_H2O = -Properties.membrane.permeability * ...
    (M_int_c.press - M_int_a.press) * water_mass_density;

%% 4. TOTAL MASS FLUX OF H2O THROUGH THE MEMBRANE

% Total mass flux of H2O
mass_flux_H2O = ( Flags.model.m.diffusion * dif_mole_flux_H2O + Flags.model.m.EOD * EOD_mole_flux_H2O ) ...
    * Properties.molar_mass.H2O + Flags.model.m.permeation * permeation_mass_flux_H2O;

%% 4. RECAST MEMBRANE MASS FLUX VECTOR INTO MASS FLUX VECTORS WITH THE ANODE AND CATHODE SPECIES ORDERS

% Create a vector containing the water and Na+ mass fluxes
Mass_fluxes_m = [mass_flux_H2O mass_flux_Na];

Mass_fluxes_a = zeros(Grid_size.x_d_num, Species.fuel.num,1);
Mass_fluxes_c = zeros(Grid_size.x_d_num, Species.oxidizer.num,1);

% Mass fluxes based on the anode and cathode species pointers
% The negative sign in front of the cathode mass flux indicates that the mass fluxes as calculated above

```

```

% are positive when flowing into the membrane from the anode side in the anode coordinate system, but
% that same mass flux flows out of the membrane and into the cathode side, which is negative in the
% cathode coordinate system.

```

```

Mass_fluxes_a(:, Species.membrane.loc_anode) = Mass_fluxes_m; % kg/(m^2 s)
Mass_fluxes_c(:, Species.membrane.loc_cathode) = -Mass_fluxes_m; % kg/(m^2 s)

```

```

end

```

```

function [mass_flux mass_density Mass_fracs charge_density] = ...
    FUNC_SCB_VALUES( Sln, Xfluxes_mass, Yfluxes_mass, Geometry, Grid_size, Species )

```

```

%% SUMMARY: FUNC_SCB_VALUES

```

```

% Purpose: This function calculates the mass density, mass fractions, charge density and total mass
% fluxes at the scalar cell boundaries for use in the continuity, N-S and species conservation equations.
% Author: Rick Stroman

```

```

%% NOTES:

```

```

% The mass and species conservation equations ensure those properties are conserved over the scalar
% cells. To evaluate the equations, we need to know the mass density, mass fractions, and total mass
% fluxes at the scalar cell boundaries. The FUNC_SCB_VALUES function calculates the properties by linear
% interpolation between cell centers (accounting for different cell sizes). The mass fluxes are the sum
% of advection, diffusion and migration mass fluxes at the boundaries. The lowest levels in each
% structure are .x and .y, which signify whether the values are at cell boundaries in the x-direction or
% y-direction.

```

```

%% 1. INITIALIZE VARIABLES STORING THE BOUNDARY PROPERTIES AND FLUXES

```

```

mass_density.x = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num );
mass_density.y = zeros(Grid_size.x_d_num , Grid_size.y_d_num + 1);

```

```

mass_flux.x = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num );
mass_flux.y = zeros(Grid_size.x_d_num , Grid_size.y_d_num + 1);

Mass_fracs.x = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num, Species.num);
Mass_fracs.y = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1, Species.num);

charge_density.x = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num );
charge_density.y = zeros(Grid_size.x_d_num , Grid_size.y_d_num + 1);

%% 2. CALCULATE PROPERTIES AT THE SCALAR CELL BOUNDARIES

% Calculate the value at each cell boundary as the linear average of the values in the two adjacent cell
% centers, weighted to account for the boundary not necessarily being halfway between the centers.

% PROPERTIES AT THE x-DIRECTION SCALAR CELL BOUNDARIES

% Calculate the value at each cell boundary, starting with the inlet and ending with the outlet. The
% values at the inlet and outlet are the average between a ghost cell and a real cell. This excludes the
% ghost cells along the electrode and membrane.

for x_d = 1 : Grid_size.x_d_num + 1

    % Distance between scalar cell centers in the x-direction
    one_over_x_step = 1 / ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d+1) );

    for y_d = 2 : Grid_size.y_d_num + 1

        % Note that a 1/2 that appears in both the numerator and denominator of the gradients below
        % has been dropped...
        mass_density.x(x_d,y_d-1) = Sln.mass_density(x_d,y_d) ...
            + Geometry.x_d_size(x_d) * ( Sln.mass_density(x_d+1,y_d) - ...
            Sln.mass_density(x_d,y_d) ) * one_over_x_step;
    end
end

```

```

charge_density.x(x_d,y_d-1) = Sln.charge_density(x_d,y_d) ...
    + Geometry.x_d_size(x_d) * ( Sln.charge_density(x_d+1,y_d) - ...
    Sln.charge_density(x_d,y_d) ) * one_over_x_step;

for s_i = 1 : Species.num

    Mass_fracs.x(x_d,y_d-1,s_i) = Sln.Mass_fracs(x_d,y_d,s_i) ...
    + Geometry.x_d_size(x_d) * ( Sln.Mass_fracs(x_d+1,y_d,s_i) - ...
    Sln.Mass_fracs(x_d,y_d,s_i) ) * one_over_x_step;

end

end

end

% PROPERTIES AT THE y-DIRECTION SCALAR CELL BOUNDARIES

% Calculate the value at each cell boundary, starting with the electrode interface and ending at the
% membrane interface. The values at the electrode and membrane interfaces are averages of a ghost cell
% and a real cell. This excludes the ghost cells along the inlet and outlet.

for x_d = 2 : Grid_size.x_d_num + 1
    for y_d = 1 : Grid_size.y_d_num + 1

        % Distance between scalar cell centers in the y-direction
        one_over_y_step = 1 / ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d+1) );

        mass_density.y(x_d-1,y_d) = Sln.mass_density(x_d,y_d) ...
            + Geometry.y_d_size(y_d) * ( Sln.mass_density(x_d,y_d+1) ...
            - Sln.mass_density(x_d,y_d) ) * one_over_y_step;

        charge_density.y(x_d-1,y_d) = Sln.charge_density(x_d,y_d) ...
            + Geometry.y_d_size(y_d) * ( Sln.charge_density(x_d,y_d+1) ...
            - Sln.charge_density(x_d,y_d) ) * one_over_y_step;
    end
end

```

```

    for s_i = 1 : Species.num

        Mass_frac.y(x_d-1,y_d,s_i) = Sln.Mass_frac(x_d,y_d,s_i) ...
            + Geometry.y_d_size(y_d) * ( Sln.Mass_frac(x_d,y_d+1,s_i) ...
            - Sln.Mass_frac(x_d,y_d,s_i) ) * one_over_y_step;

    end

end

end

%% 3. CALCULATE MASS FLUXES AT THE SCALAR CELL BOUNDARIES

% Sum the advection (rho*v), diffusion and migration mass fluxes. The sums of diffusion and migration
% are stored in the Xfluxes and Yfluxes arrays, having been calculated beforehand by another function.

% MASS FLUXES AT THE SCALAR CELL BOUNDARIES IN THE x-DIRECTION

for x_d = 1 : Grid_size.x_d_num + 1 % Includes fluxes at inlet and exit
    for y_d = 2 : Grid_size.y_d_num + 1
        mass_flux.x(x_d,y_d-1) = mass_density.x(x_d,y_d-1) * Sln.x_vel(x_d,y_d) ...
            + sum( Xfluxes_mass(x_d,y_d-1,:) );
    end
end

% MASS FLUXES AT THE SCALAR CELL BOUNDARIES IN THE y-DIRECTION

for x_d = 2 : Grid_size.x_d_num + 1
    for y_d = 1 : Grid_size.y_d_num + 1
        mass_flux.y(x_d-1,y_d) = mass_density.y(x_d-1,y_d) * Sln.y_vel(x_d,y_d) ...
            + sum( Yfluxes_mass(x_d-1,y_d,:) );
    end
end
end

```

```
end
```

```
function [x_dir_x_vel, y_dir_y_vel] = FUNC_VCB_VALUES( Sln, Grid_size )
```

```
%% SUMMARY: FUNC_VCB_VALUES
```

```
% Purpose: Calculate the velocities at the boundaries of the velocity cells as a linear average of  
% the velocities in the adjacent cell centers.
```

```
% Author: Rick Stroman
```

```
%% 1. INITIALIZE VARIABLES
```

```
x_dir_x_vel = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num);
```

```
y_dir_y_vel = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num);
```

```
%% 2. CALCULATE PROPERTIES AT THE BOUNDARIES
```

```
for x_d = 2 : Grid_size.x_d_num + 2 % The additional two cells are guard cells for BC's
```

```
    for y_d = 2 : Grid_size.y_d_num + 1 % The additional two cells are guard cells for BC's
```

```
        x_r = x_d - 1;  y_r = y_d - 1;
```

```
        x_dir_x_vel(x_r,y_r) = 0.5 * ( Sln.x_vel(x_d,y_d) + Sln.x_vel(x_d-1,y_d) );
```

```
        y_dir_y_vel(x_r,y_r) = 0.5 * ( Sln.y_vel(x_d,y_d) + Sln.y_vel(x_d,y_d-1) );
```

```
    end
```

```
end
```

```

function [Mass_fluxes, delta_phi] = FUNC_REACTION_FLUXES( phi_e, Interface, Srxn, Properties, ...
                                                         Grid_size, Species, Constants)

%% SUMMARY: FUNC_REACTION_FLUXES
% Purpose: Compute the mass fluxes of species from electrode.
% Author: Rick Stroman

%% NOTES:

% (1) Mass fluxes are returned in units kg/(m^2 s) assuming the temperature is in K, mole density in
%     kmol/m^3, and electric potential in V.
%
% (2) The rate expression for each reaction follows the form laid out in references [1] pg 210 and
% [2] page 2388. For the chemical reactions, there are simply forward and reverse reaction rate
% constants and concentration dependencies. For the charge transfer (electrochemical) reactions,
% there are also activation energy barriers created by the electric potential difference between the
% electrode and solution.
%
% (4) The electric potential of the electrode and interface are passed into this function as vectors
%     covering the whole length of the channel, so the function returns a matrix of species mass fluxes,
%     one species indexed vector for each x-discretization.
%
% (5) The electron transfer reactions must be reversible to correctly predict the open circuit
%     voltage of the cell... even if the reverse reaction has a very, very slow rate and the overall
%     reaction is nearly irreversible.

%% 1. SET UP AND INITIALIZE SOME NECESSARY SHARED PARAMETERS

x_d_range      = 1 : Grid_size.x_d_num;
species_range  = 1 : Species.num;

% Initialize the mass and mole fluxes of each species from the electrode to the interface
Mole_fluxes = zeros(Grid_size.x_d_num , Species.num);
Mass_fluxes = zeros(Grid_size.x_d_num , Species.num);

```

```

reaction_rate = zeros(Grid_size.x_d_num, 4);

% Make a vector containing the electrode potential at each point in x-direction
phi_e = phi_e * ones(Grid_size.x_d_num,1);

% Calculate the potential drop across the electrode-solution interface
delta_phi = phi_e - Interface.elec_pot;

%% 2. CALCULATE THE RATES AND FLUXES

for r_i = 1:length(Srxn.rxn) % Cycle through all of the reactions, one at a time

    % Create the concentration dependence terms for the anodic and cathode directions, if the are specified
    % by the user.  Otherwise the default dependence is 1, i.e. no dependence.

    if ~isempty(Srxn.rxn(r_i).conc_dependence_r)
        Conc_dependence_r = prod(Interface.Mole_densities(x_d_range,Srxn.rxn(r_i).conc_dependence_r),2);
    else
        Conc_dependence_r = ones(Grid_size.x_d_num , 1);
    end

    if ~isempty(Srxn.rxn(r_i).conc_dependence_f)
        Conc_dependence_f = prod(Interface.Mole_densities(x_d_range,Srxn.rxn(r_i).conc_dependence_f),2);
    else
        Conc_dependence_f = ones(Grid_size.x_d_num , 1);
    end

    rate_f = Srxn.rxn(r_i).k_f * Conc_dependence_f(x_d_range,1) .* exp( Srxn.rxn(r_i).e_rds * ...
        Srxn.rxn(r_i).beta_f * Constants.FoRT * delta_phi(x_d_range,1));
    rate_r = Srxn.rxn(r_i).k_r * Conc_dependence_r(x_d_range,1) .* exp(-Srxn.rxn(r_i).e_rds * ...
        Srxn.rxn(r_i).beta_r * Constants.FoRT * delta_phi(x_d_range,1));

    reaction_rate = rate_f - rate_r;

```

```

% TOTAL REACTION MOLE FLUXES TO THE SURFACE

% Calculate the mole fluxes at each location down the channel and add them to the values from previous
% reactions to give the total so far.

Mole_fluxes(x_d_range,species_range) = Mole_fluxes(x_d_range,species_range) ...
+ Srxn.rxn(r_i).active * reaction_rate(x_d_range,1) * Srxn.rxn(r_i).Reaction_stoich; % kmol/(m^2 s)

end

%% 4. ACCOUNT FOR ROUGHNESS OF CATALYST SURFACE

Mole_fluxes = Srxn.param.area_ratio * Mole_fluxes;

% 5. CONVERT THE TOTAL MOLE FLUXES INTO MASS FLUXES FOR OUTPUT FROM THE FUNCTION

for x_d = 1:Grid_size.x_d_num
    Mass_fluxes(x_d,species_range) = Mole_fluxes(x_d,species_range)' .* Properties.Molar_mass; % kg/(m^2 s)
end

%% REFERENCES
% [1] Newman, J. S. and K. E. Thomas-Alyea (2004). Electrochemical systems. Hoboken, N.J., J. Wiley.
% [2] Kee, R. J., H. Y. Zhu, et al. (2005). "Solid-oxide fuel cells with hydrocarbon fuels."
% Proceedings of the Combustion Institute 30: 2379-2404.

end

```

```

function [d_SV] = FUNC_RESIDUALS(State, Yfluxes, Xfluxes, Membrane_mass_flux, SCB, VCB, ...
    Pointer, Flags, Geometry, Grid_size, Species, Properties, Constants) %#codegen

d_SV = zeros(Grid_size.state_vars_num,1);

% Renaming the variables isn't very efficient, but the code below would be almost
% unreadable with the extra text.
A_int    = State.A_int;
Asln     = State.Asln;
M_int_a  = State.M_int_a;
M_int_c  = State.M_int_c;
Csln     = State.Csln;
C_int    = State.C_int;

%% 1. SOLVE FOR PRESSURE USING CONTINUITY

% CONTINUITY - SOLVE RESIDUALS ASSOCIATED WITH PRESSURE  del dot (rho*v) = - d_rho/d_t -----

% Note that this section uses "real" indices x_r and y_r which exclude ghost cells, because it balances
% mass over real cells and does not need the ghost cells.  The fluxes between ghost cells and real cells
% at the periphery were calculated beforehand and are included in SCB.

for x_r = 1 : Grid_size.x_d_num

    for y_r = 1 : Grid_size.y_d_num

        % NOTE: Written as d rho / dt = - ( d J_x / dx + d J_y / dy ) where J is the total mass flux
        % (advection, migration and diffusion).  Example: For the first x-discretization, we want the mass
        % fluxes in and out in the x-direction, which are at x_d = 1 and x_d = 2.  Since x_d starts at x_d =
        % 2, this is what we get below.

        % Continuity in the bulk cells

        % Asln

```

```

d_SV(Pointer.asln.press(x_r,y_r)) = ... % This is d rho / d t
( ...
... % x-direction mass flux due to advection, migration and diffusion (out) - (in)
- ( SCB.asln.mass_flux.x(x_r+1,y_r) - SCB.asln.mass_flux.x(x_r,y_r) ) / Geometry.x_d_size(x_r+1)...
... % y-direction mass flux due to advection, migration and diffusion (upwind) (out)-(in)
- ( SCB.asln.mass_flux.y(x_r,y_r+1) - SCB.asln.mass_flux.y(x_r,y_r) ) / Geometry.y_d_size(y_r+1) );

% Csln
d_SV(Pointer.csln.press(x_r,y_r)) = ... % This is d rho / d t
( ...
... % x-direction mass flux due to advection, migration and diffusion (out) - (in)
- ( SCB.csln.mass_flux.x(x_r+1,y_r) - SCB.csln.mass_flux.x(x_r,y_r) ) / Geometry.x_d_size(x_r+1)...
... % y-direction mass flux due to advection, migration and diffusion (upwind) (out)-(in)
- ( SCB.csln.mass_flux.y(x_r,y_r+1) - SCB.csln.mass_flux.y(x_r,y_r) ) / Geometry.y_d_size(y_r+1) );

```

end

```
% Continuity at the membrane interfaces
```

```
% Anode solution-membrane interface.
```

```

% (total advection mass flux in) + (total migration + diffusion mass fluxes in)
% - (total membrane mass flux out) = 0
d_SV(Pointer.m_int_a.press(x_r)) = SCB.asln.mass_flux.y(x_r,Grid_size.y_d_num+1) ...
- Membrane_mass_flux.am(x_r);

```

```
% Membrane - cathode solution interface.
```

```

% (total membrane mass flux in) - (total advection mass flux out)
% - (migration + diffusion mass fluxes out) = 0
% Note that in this coordinate system the membrane flux is negative, because it is leaving the membrane!
d_SV(Pointer.m_int_c.press(x_r)) = SCB.csln.mass_flux.y(x_r,Grid_size.y_d_num+1) ...
- Membrane_mass_flux.cm(x_r);

```

end

```
% Store d_rho/dt for each bulk cell in a local variable for use later in the momentum and species
```

```

% conservation equations.  First initialize these local variables to zero.  Include extra rows and
% columns of zeros at the edges for the membrane interface ghost cells and outlet ghost cells... this way
% when d_rho/dt is found at the centers of the velocity cells near the edge of the model domain, there is
% a second value for the average and the code won't crash.

d_rho_dt.asln = zeros(Grid_size.x_d_num+1,Grid_size.y_d_num);
d_rho_dt.csln = zeros(Grid_size.x_d_num+1,Grid_size.y_d_num);

d_rho_dt.asln(1:Grid_size.x_d_num,:) = d_SV(Pointer.asln.press(:,:));
d_rho_dt.csln(1:Grid_size.x_d_num,:) = d_SV(Pointer.csln.press(:,:));

% Make d_rho_dt in the ghost cells at outlet the same as the last of the real cells.
d_rho_dt.asln(Grid_size.x_d_num+1,:) = d_SV(Pointer.asln.press(Grid_size.x_d_num,:));
d_rho_dt.csln(Grid_size.x_d_num+1,:) = d_SV(Pointer.csln.press(Grid_size.x_d_num,:));

%% 2. SOLVE FOR VELOCITY USING THE 2D N-S EQUATIONS

% The 2D Navier-Stokes equation is evaluated over each velocity cell to solve for the velocity in
% that cell.

% CALCULATE THE SHEAR STRESS DERIVATIVES IN EACH REAL CELL

% Start by initializing the shear stress derivatives
d2_vx_d2_x_asln = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
d2_vx_d2_x_csln = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
d2_vx_d2_y_asln = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
d2_vx_d2_y_csln = zeros(Grid_size.x_d_num, Grid_size.y_d_num);

d2_vy_d2_y_asln = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
d2_vy_d2_y_csln = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
d2_vy_d2_x_asln = zeros(Grid_size.x_d_num, Grid_size.y_d_num);
d2_vy_d2_x_csln = zeros(Grid_size.x_d_num, Grid_size.y_d_num);

% Evaluate the shear stress derivatives

```

```

% Note that the last set of shear stress derivatives for v_y along the membrane are calculated separately
% because we do not have the necessary values for v_y beyond the last cell to calculate d^2 v_y / dy^2.
% Eventually I will implement a scheme to estimate what the value in the missing cell would be, so this
% derivative can be computed... but for now it is small and assumed to be zero.

```

```

for x_d = 2 : Grid_size.x_d_num + 1 % Run through all real cells between the inlet and outlet in the DCS

```

```

    x_r = x_d - 1;

```

```

    for y_d = 2 : Grid_size.y_d_num + 1

```

```

        % Run through all real cells between the electrode and membrane in the DCS

```

```

            y_r = y_d-1;

```

```

            % Calculate the 2nd derivatives of vx -----

```

```

            d2_vx_d2_y_asln(x_r,y_r) = Geometry.y_d_size(y_d)^-1 * ...
            ( ( Asln.x_vel(x_d,y_d+1) - Asln.x_vel(x_d,y_d) ) ...
              / ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d+1) ) ) ...
            - ( Asln.x_vel(x_d,y_d) - Asln.x_vel(x_d,y_d-1) ) ...
              / ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d-1) ) ) );

```

```

            d2_vx_d2_y_csln(x_r,y_r) = Geometry.y_d_size(y_d)^-1 * ...
            ( ( Csln.x_vel(x_d,y_d+1) - Csln.x_vel(x_d,y_d) ) ...
              / ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d+1) ) ) ...
            - ( Csln.x_vel(x_d,y_d) - Csln.x_vel(x_d,y_d-1) ) ...
              / ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d-1) ) ) );

```

```

            d2_vx_d2_x_asln(x_r,y_r) = Geometry.x_d_size(x_d)^-1 * ...
            ( ( Asln.x_vel(x_d+1,y_d) - Asln.x_vel(x_d,y_d) ) ...
              / ( 0.5 * ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d+1) ) ) ...
            - ( Asln.x_vel(x_d,y_d) - Asln.x_vel(x_d-1,y_d) ) ...
              / ( 0.5 * ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d-1) ) ) );

```

```

/ ( 0.5 * ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d-1) ) ) );

d2_vx_d2_x_csln(x_r,y_r) = Geometry.x_d_size(x_d)^-1 * ...
( ( Csln.x_vel(x_d+1,y_d) - Csln.x_vel(x_d,y_d) ) ...
/ ( 0.5 * ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d+1) ) ) ...
- ( Csln.x_vel(x_d,y_d) - Csln.x_vel(x_d-1,y_d) ) ...
/ ( 0.5 * ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d-1) ) ) );

% Calculate the 2nd derivatives of vy -----

d2_vy_d2_x_asln(x_r,y_r) = Geometry.x_d_size(x_d)^-1 * ...
( ( Asln.y_vel(x_d+1,y_d) - Asln.y_vel(x_d,y_d) ) ...
/ ( 0.5 * (Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d+1) ) ) ...
- ( Asln.y_vel(x_d,y_d) - Asln.y_vel(x_d-1,y_d) ) ...
/ ( 0.5 * (Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d-1) ) ) );

d2_vy_d2_x_csln(x_r,y_r) = Geometry.x_d_size(x_d)^-1 * ...
( ( Csln.y_vel(x_d+1,y_d) - Csln.y_vel(x_d,y_d) ) ...
/ ( 0.5 * (Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d+1) ) ) ...
- ( Csln.y_vel(x_d,y_d) - Csln.y_vel(x_d-1,y_d) ) ...
/ ( 0.5 * (Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d-1) ) ) );

end

for y_d = 2 : Grid_size.y_d_num
% Run through all real cells between the electrode and membrane, in the DCS, except the membrane
% interface.
y_r = y_d-1;

% Calculate the 2nd derivatives of vy -----

d2_vy_d2_y_asln(x_r,y_r) = Geometry.y_d_size(y_d)^-1 * ...
( ( Asln.y_vel(x_d,y_d+1) - Asln.y_vel(x_d,y_d) ) ...
/ ( 0.5 * ( Geometry.y_d_size(y_d+1) + Geometry.y_d_size(y_d) ) ) ...

```

```

- ( Asln.y_vel(x_d,y_d) - Asln.y_vel(x_d,y_d-1) ) ...
/ ( 0.5 * ( Geometry.y_d_size(y_d-1) + Geometry.y_d_size(y_d) ) ) );

d2_vy_d2_y_csln(x_r,y_r) = Geometry.y_d_size(y_d)^-1 * ...
( ( Csln.y_vel(x_d,y_d+1) - Csln.y_vel(x_d,y_d) ) ...
/ ( 0.5 * ( Geometry.y_d_size(y_d+1) + Geometry.y_d_size(y_d) ) ) ...
- ( Csln.y_vel(x_d,y_d) - Csln.y_vel(x_d,y_d-1) ) ...
/ ( 0.5 * ( Geometry.y_d_size(y_d-1) + Geometry.y_d_size(y_d) ) ) );

```

end

end

% MOMENTUM (N-S) - SOLVE FOR RESIDUALS ASSOCIATED WITH v_x -----

% Initialize advection values at boundaries between cells

```

Adv.x.asln.x_vel      = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1);
Adv.x.asln.y_vel      = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1);
Adv.x.asln.mass_density = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1);

Adv.x.csln.x_vel      = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1);
Adv.x.csln.y_vel      = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1);
Adv.x.csln.mass_density = zeros(Grid_size.x_d_num, Grid_size.y_d_num + 1);

Adv.y.asln.x_vel      = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num - 1);
Adv.y.asln.y_vel      = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num - 1);
Adv.y.asln.mass_density = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num - 1);

Adv.y.csln.x_vel      = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num - 1);
Adv.y.csln.y_vel      = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num - 1);
Adv.y.csln.mass_density = zeros(Grid_size.x_d_num + 1, Grid_size.y_d_num - 1);

```

% Written as $d v_x / d t = (1 / \rho) * \{ -d \text{momentum_flux_x} / dx - d \text{momentum_flux_x} / dy -$

```

% d P / dx + d [mu * ( d v_x / dy + d v_y / dx)] / dy + F_elec - v_x * d rho / dt }

for x_d = 2 : Grid_size.x_d_num + 1 % All real v_x cells between the inlet and outlet, in the DCS
    x_r = x_d - 1; % Corresponding position in the RCS

for y_d = 2 : Grid_size.y_d_num + 1 % Real v_x cells between the electrode and membrane, in the DCS
    y_r = y_d - 1; % Corresponding position in the RCS

% Linearly interpolate between cell centers to estimate v_x, v_y and the mass density on the velocity
% cell boundaries for use in the momentum advection terms in the N-S equation. These are used to
% determine how fast x-momentum is carried across the cell boundaries by v_y and vice versa.

% y-velocity carrying x-momentum in the y-direction
Adv.x.asln.y_vel(x_r,y_r) = 0.5 * ( Asln.y_vel(x_d,y_d-1) + Asln.y_vel(x_d+1,y_d-1) );
Adv.x.asln.y_vel(x_r,y_r+1) = 0.5 * ( Asln.y_vel(x_d,y_d) + Asln.y_vel(x_d+1,y_d) );

Adv.x.csln.y_vel(x_r,y_r) = 0.5 * ( Csln.y_vel(x_d,y_d-1) + Csln.y_vel(x_d+1,y_d-1) );
Adv.x.csln.y_vel(x_r,y_r+1) = 0.5 * ( Csln.y_vel(x_d,y_d) + Csln.y_vel(x_d+1,y_d) );

% x-velocity producing the x-momentum which is carried in the y-direction by v_y
Adv.x.asln.x_vel(x_r,y_r) = Asln.x_vel(x_d,y_d) - Geometry.y_d_size(y_d) * ...
    (Asln.x_vel(x_d,y_d) - Asln.x_vel(x_d,y_d-1)) ...
    / (Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d-1));
Adv.x.asln.x_vel(x_r,y_r+1) = Asln.x_vel(x_d,y_d+1) - Geometry.y_d_size(y_d+1) * ...
    (Asln.x_vel(x_d,y_d+1) - Asln.x_vel(x_d,y_d) ) ...
    / (Geometry.y_d_size(y_d+1) + Geometry.y_d_size(y_d) );

Adv.x.csln.x_vel(x_r,y_r) = Csln.x_vel(x_d,y_d) - Geometry.y_d_size(y_d) * ...
    (Csln.x_vel(x_d,y_d) - Csln.x_vel(x_d,y_d-1)) ...
    / (Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d-1));
Adv.x.csln.x_vel(x_r,y_r+1) = Csln.x_vel(x_d,y_d+1) - Geometry.y_d_size(y_d+1) * ...
    (Csln.x_vel(x_d,y_d+1) - Csln.x_vel(x_d,y_d) ) ...
    / (Geometry.y_d_size(y_d+1) + Geometry.y_d_size(y_d) );
% Mass density at the velocity cell boundaries in the y-direction, used to calculate the
% x-momentum at those boundaries.

```

```

Adv.x.asln.mass_density(x_r,y_r) = 0.25 * (Asln.mass_density(x_d,y_d)+ ...
  Asln.mass_density(x_d,y_d-1)+Asln.mass_density(x_d+1,y_d)+Asln.mass_density(x_d+1,y_d-1));
Adv.x.asln.mass_density(x_r,y_r+1) = 0.25 * (Asln.mass_density(x_d,y_d)+ ...
  Asln.mass_density(x_d,y_d+1)+Asln.mass_density(x_d+1,y_d)+Asln.mass_density(x_d+1,y_d+1));

Adv.x.csln.mass_density(x_r,y_r) = 0.25 * (Csln.mass_density(x_d,y_d)+ ...
  Csln.mass_density(x_d,y_d-1)+Csln.mass_density(x_d+1,y_d)+Csln.mass_density(x_d+1,y_d-1));
Adv.x.csln.mass_density(x_r,y_r+1) = 0.25 * (Csln.mass_density(x_d,y_d)+ ...
  Csln.mass_density(x_d,y_d+1)+Csln.mass_density(x_d+1,y_d)+Csln.mass_density(x_d+1,y_d+1));

% Evaluate the N-S equation

% Anode Side
d_SV(Pointer.asln.x_vel(x_r,y_r)) = SCB.asln.mass_density.x(x_r+1,y_r)^-1 * ...
  ( ...
  ... % x-direction advection, migration and diffusion of x-direction momentum (center differenced
  ... because total mass flux uses density at boundary) (out) - (in)
  - (abs(VCB.x.asln.x_vel(x_r+1,y_r)) * Asln.mass_density(x_d+1,y_d) * VCB.x.asln.x_vel(x_r+1,y_r)...
  - abs( VCB.x.asln.x_vel(x_r,y_r) ) * Asln.mass_density(x_d,y_d) * VCB.x.asln.x_vel(x_r,y_r) ) ...
  / (0.5 * (Geometry.x_d_size(x_d)+Geometry.x_d_size(x_d+1))) ...
  ... % y-direction advection, migration and diffusion of x-direction momentum
  ... (center differenced) (out) - (in)
  - (Adv.x.asln.y_vel(x_r,y_r+1) * Adv.x.asln.x_vel(x_r,y_r+1) * Adv.x.asln.mass_density(x_r,y_r+1)...
  - Adv.x.asln.y_vel(x_r,y_r) * Adv.x.asln.x_vel(x_r,y_r) * Adv.x.asln.mass_density(x_r,y_r) ) ...
  / Geometry.y_d_size(y_d) ...
  ... % x-direction normal forces due to pressure on the upwind and downwind faces
  ... (upwind differenced) (downstream face) - (upstream face)
  - (Asln.press(x_d+1,y_d) - Asln.press(x_d,y_d)) ...
  / ( 0.5 * ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d+1) ) ) ...
  ... % x-direction shear forces due to viscosity
  ... (center differenced by virtue of second derivatives)
  + Properties.anode.viscosity * ( d2_vx_d2_y_asln(x_r,y_r) + d2_vx_d2_x_asln(x_r,y_r) ) ...
  ... % Body force due to electric field acting on ions in the solution (center differenced)
  - SCB.asln.charge_density.x(x_r+1,y_r) * ( Asln.elec_pot(x_d+1, y_d) - Asln.elec_pot(x_d,y_d) ) ...
  / ( 0.5 * (Geometry.x_d_size(x_d+1) + Geometry.x_d_size(x_d)) ) ...

```

```

... % d rho / d t term from the LHS
- Asln.x_vel(x_d,y_d) * 0.5 * ( d_rho_dt.asln(x_r,y_r) + d_rho_dt.asln(x_r+1,y_r) ) ...
);

```

```

% Cathode Side

```

```

d_SV(Pointer.csln.x_vel(x_r,y_r)) = SCB.csln.mass_density.x(x_r+1,y_r)^-1 * ...
( ...
... % x-direction advection, migration and diffusion of x-direction momentum
... (center differenced because total mass flux uses density at boundary) (out) - (in)
- (abs( VCB.x.csln.x_vel(x_r+1,y_r)) * Csln.mass_density(x_d+1,y_d) * VCB.x.csln.x_vel(x_r+1,y_r)...
- abs( VCB.x.csln.x_vel(x_r,y_r)) * Csln.mass_density(x_d,y_d) * VCB.x.csln.x_vel(x_r,y_r) ) ...
/ (0.5 * (Geometry.x_d_size(x_d)+Geometry.x_d_size(x_d+1))) ...
... % y-direction advection, migration and diffusion of x-direction momentum
... (center differenced) (out) - (in)
- (Adv.x.csln.y_vel(x_r,y_r+1) * Adv.x.csln.x_vel(x_r,y_r+1) * Adv.x.csln.mass_density(x_r,y_r+1)...
- Adv.x.csln.y_vel(x_r,y_r) * Adv.x.csln.x_vel(x_r,y_r) * Adv.x.csln.mass_density(x_r,y_r) ) ...
/ Geometry.y_d_size(y_d) ...
... % x-direction normal forces due to pressure on the upwind and downwind faces
... (upwind differenced) (downstream face) - (upstream face)
- ( Csln.press(x_d+1,y_d) - Csln.press(x_d,y_d) ) ...
/ ( 0.5 * ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d+1) ) ) ...
... % x-direction shear forces due to viscosity
... (center differenced by virtue of second derivatives)
+ Properties.cathode.viscosity * ( d2_vx_d2_y_csln(x_r,y_r) + d2_vx_d2_x_csln(x_r,y_r) ) ...
... % Body force due to electric field acting on ions in the solution (center differenced)
- SCB.csln.charge_density.x(x_r+1,y_r) * ( Csln.elec_pot(x_d+1, y_d) - Csln.elec_pot(x_d,y_d) ) ...
/ ( 0.5 * ( Geometry.x_d_size(x_d+1) + Geometry.x_d_size(x_d) ) ) ...
... % d rho / d t term from the LHS
- Csln.x_vel(x_d,y_d) * 0.5 * ( d_rho_dt.csln(x_r,y_r) + d_rho_dt.csln(x_r+1,y_r) )...
);

```

```

end

```

```

end

```

```

for x_d = 2 : Grid_size.x_d_num + 1 % All real v_x cells between the inlet and outlet, in the DCS

x_r = x_d - 1;

for y_d = 2 : Grid_size.y_d_num % All real v_x cells between the electrode and membrane, in the DCS,
% EXCEPT the cells along the membrane, which have thier own momentum
% equations

y_r = y_d - 1; % Corresponding position in the RCS

% MOMENTUM CONSERVATION (N-S) - SOLVE FOR RESIDUALS ASSOCIATED WITH v_y -----

% Written as d v_y / d t = ( 1 / rho ) * { - d momentum_flux_y / dx - d momentum_flux_y / dy
% - d P / dy + d [mu * ( d v_x / dy + d v_y / dx)] / dx + F_elec - v_y * d rho / dt }

Adv.y.asln.x_vel(x_r,y_r) = 0.5 * ( Asln.x_vel(x_d-1,y_d) + Asln.x_vel(x_d-1,y_d+1) );
Adv.y.asln.x_vel(x_r+1,y_r) = 0.5 * ( Asln.x_vel(x_d, y_d) + Asln.x_vel(x_d, y_d+1) );

Adv.y.csln.x_vel(x_r,y_r) = 0.5 * ( Csln.x_vel(x_d-1,y_d) + Csln.x_vel(x_d-1,y_d+1) );
Adv.y.csln.x_vel(x_r+1,y_r) = 0.5 * ( Csln.x_vel(x_d, y_d) + Csln.x_vel(x_d, y_d+1) );

Adv.y.asln.y_vel(x_r,y_r) = Asln.y_vel(x_d, y_d) - Geometry.x_d_size(x_d) ...
* ( Asln.y_vel(x_d,y_d) - Asln.y_vel(x_d-1,y_d) ) / ...
( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d-1) );
Adv.y.asln.y_vel(x_r+1,y_r) = Asln.y_vel(x_d+1,y_d) - Geometry.x_d_size(x_d+1) ...
* ( Asln.y_vel(x_d+1,y_d) - Asln.y_vel(x_d,y_d) ) ...
/ ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d+1) );

Adv.y.csln.y_vel(x_r,y_r) = Csln.y_vel(x_d, y_d) - Geometry.x_d_size(x_d) ...
* ( Csln.y_vel(x_d,y_d) - Csln.y_vel(x_d-1,y_d) ) ...
/ ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d-1) );
Adv.y.csln.y_vel(x_r+1,y_r) = Csln.y_vel(x_d+1,y_d) - Geometry.x_d_size(x_d+1) ...
* ( Csln.y_vel(x_d+1,y_d) - Csln.y_vel(x_d,y_d) ) ...

```

```

/ ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d+1) );

Adv.y.asln.mass_density(x_r,y_r) = 0.25 * ( Asln.mass_density(x_d,y_d) ...
+ Asln.mass_density(x_d,y_d+1) + Asln.mass_density(x_d-1,y_d) + Asln.mass_density(x_d-1,y_d-1));
Adv.y.asln.mass_density(x_r+1,y_r) = 0.25 * ( Asln.mass_density(x_d,y_d) ...
+ Asln.mass_density(x_d,y_d+1) + Asln.mass_density(x_d+1,y_d) + Asln.mass_density(x_d+1,y_d+1));

Adv.y.csln.mass_density(x_r,y_r) = 0.25 * ( Csln.mass_density(x_d,y_d) ...
+ Csln.mass_density(x_d,y_d+1) + Csln.mass_density(x_d-1,y_d) + Csln.mass_density(x_d-1,y_d-1));
Adv.y.csln.mass_density(x_r+1,y_r) = 0.25 * ( Csln.mass_density(x_d,y_d) ...
+ Csln.mass_density(x_d,y_d+1) + Csln.mass_density(x_d+1,y_d) + Csln.mass_density(x_d+1,y_d+1));

% Evaluate the N-S equation

% Anode Side
d_SV(Pointer.asln.y_vel(x_r,y_r)) = SCB.asln.mass_density.y(x_r,y_r+1)^-1 * ...
( ...
... % y-direction advection, migration and diffusion of y-direction momentum
... (center differenced) (out) - (in)
- ( abs( VCB.y.asln.y_vel(x_r,y_r+1) ) * Asln.mass_density(x_d,y_d+1) ...
* VCB.y.asln.y_vel(x_r,y_r+1) - abs( VCB.y.asln.y_vel(x_r,y_r) ) ...
* Asln.mass_density(x_d,y_d) * VCB.y.asln.y_vel(x_r,y_r) ) ...
/ (0.5 * (Geometry.y_d_size(y_d)+Geometry.y_d_size(y_d+1))) ...
... % x-direction advection, migration and diffusion of y-direction momentum
... (upwind differenced) (out) - (in)
- ( Adv.y.asln.x_vel(x_r+1,y_r) * Adv.y.asln.y_vel(x_r+1,y_r) ...
* Adv.y.asln.mass_density(x_r+1,y_r) - Adv.y.asln.x_vel(x_r, y_r) ...
* Adv.y.asln.y_vel(x_r, y_r) * Adv.y.asln.mass_density(x_r, y_r) ) ...
/ Geometry.x_d_size(x_d) ...
... % y-direction normal forces due to pressure on the upwind and downwind faces
... (center differenced)
- ( Asln.press(x_d,y_d+1) - Asln.press(x_d,y_d) ) ...
/ ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d+1)) ) ...
... % y-direction shear forces due to viscosity
+ Properties.anode.viscosity * ( d2_vy_d2_y_asln(x_r,y_r) + d2_vy_d2_x_asln(x_r,y_r) ) ...

```

```

... % Body force due to electric field acting on ions in the solution (center differenced)
- SCB.asln.charge_density.y(x_r,y_r+1) ...
  * ( Asln.elec_pot(x_d, y_d+1) - Asln.elec_pot(x_d,y_d) ) ...
  / ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d+1) ) ) ...
... % d rho / d t term from the LHS
- Asln.y_vel(x_d,y_d) * 0.5 * ( d_rho_dt.asln(x_r,y_r) + d_rho_dt.asln(x_r,y_r+1) ) ...
);

% Cathode Side
d_SV(Pointer.csln.y_vel(x_r,y_r)) = SCB.csln.mass_density.y(x_r,y_r+1)^-1 * ...
( ...
... % y-direction advection, migration and diffusion of y-direction momentum (out) - (in)
- ( abs( VCB.y.csln.y_vel(x_r,y_r+1) ) * Csln.mass_density(x_d,y_d+1) ...
  * VCB.y.csln.y_vel(x_r,y_r+1) - abs( VCB.y.csln.y_vel(x_r,y_r) ) ...
  * Csln.mass_density(x_d,y_d) * VCB.y.csln.y_vel(x_r,y_r) ) ...
  / (0.5 * (Geometry.y_d_size(y_d)+Geometry.y_d_size(y_d+1))) ...
... % x-direction advection, migration and diffusion of y-direction momentum (out) - (in)
- ( Adv.y.csln.x_vel(x_r+1,y_r) * Adv.y.csln.y_vel(x_r+1,y_r) ...
  * Adv.y.csln.mass_density(x_r+1,y_r) - Adv.y.csln.x_vel(x_r, y_r) ...
  * Adv.y.csln.y_vel(x_r, y_r) * Adv.y.csln.mass_density(x_r, y_r) ) ...
  / Geometry.x_d_size(x_d) ...
... % y-direction normal forces due to pressure on the upwind and downwind faces
- ( Csln.press(x_d,y_d+1) - Csln.press(x_d,y_d) ) ...
  / ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d+1)) ) ...
... % y-direction shear forces due to viscosity
+ Properties.cathode.viscosity * ( d2_vy_d2_y_csln(x_r,y_r) + d2_vy_d2_x_csln(x_r,y_r) ) ...
... % Body force due to electric field acting on ions in the solution
- SCB.csln.charge_density.y(x_r,y_r+1) ...
  * ( Csln.elec_pot(x_d, y_d+1) - Csln.elec_pot(x_d,y_d) ) ...
  / ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d+1) ) ) ...
... % d rho / d t term from the LHS
- Csln.y_vel(x_d,y_d) * 0.5 * ( d_rho_dt.csln(x_r,y_r) + d_rho_dt.csln(x_r,y_r+1) ) ...
);

end

```

```

y_d = Grid_size.y_d_num+1;

Adv.y.asln.x_vel(x_r,y_r)    = 0.5 * ( Asln.x_vel(x_d-1,y_d) + Asln.x_vel(x_d-1,y_d+1) );
Adv.y.asln.x_vel(x_r+1,y_r) = 0.5 * ( Asln.x_vel(x_d,  y_d) + Asln.x_vel(x_d,  y_d+1) );

Adv.y.asln.y_vel(x_r,y_r)    = Asln.y_vel(x_d,  y_d) - Geometry.x_d_size(x_d)    ...
  * ( Asln.y_vel(x_d,y_d) - Asln.y_vel(x_d-1,y_d) ) ...
  / ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d-1) );
Adv.y.asln.y_vel(x_r+1,y_r) = Asln.y_vel(x_d+1,y_d) - Geometry.x_d_size(x_d+1) ...
  * ( Asln.y_vel(x_d+1,y_d) - Asln.y_vel(x_d,y_d) ) ...
  / ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d+1) );

Adv.y.asln.mass_density(x_r,y_r)    = 0.25 * ( Asln.mass_density(x_d,y_d) ...
  + Asln.mass_density(x_d,y_d+1) + Asln.mass_density(x_d-1,y_d) + Asln.mass_density(x_d-1,y_d-1) );
Adv.y.asln.mass_density(x_r+1,y_r) = 0.25 * ( Asln.mass_density(x_d,y_d) ...
  + Asln.mass_density(x_d,y_d+1) + Asln.mass_density(x_d+1,y_d) + Asln.mass_density(x_d+1,y_d+1) );

Adv.y.csln.x_vel(x_r,y_r)    = 0.5 * ( Csln.x_vel(x_d-1,y_d) + Csln.x_vel(x_d-1,y_d+1) );
Adv.y.csln.x_vel(x_r+1,y_r) = 0.5 * ( Csln.x_vel(x_d,  y_d) + Csln.x_vel(x_d,  y_d+1) );

Adv.y.csln.y_vel(x_r,y_r)    = Csln.y_vel(x_d,  y_d) - Geometry.x_d_size(x_d)    ...
  * ( Csln.y_vel(x_d,y_d) - Csln.y_vel(x_d-1,y_d) ) ...
  / ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d-1) );
Adv.y.csln.y_vel(x_r+1,y_r) = Csln.y_vel(x_d+1,y_d) - Geometry.x_d_size(x_d+1) ...
  * ( Csln.y_vel(x_d+1,y_d) - Csln.y_vel(x_d,y_d) ) ...
  / ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d+1) );

Adv.y.csln.mass_density(x_r,y_r)    = 0.25 * ( Csln.mass_density(x_d,y_d) ...
  + Csln.mass_density(x_d,y_d+1) + Csln.mass_density(x_d-1,y_d) + Csln.mass_density(x_d-1,y_d-1) );
Adv.y.csln.mass_density(x_r+1,y_r) = 0.25 * ( Csln.mass_density(x_d,y_d) ...
  + Csln.mass_density(x_d,y_d+1) + Csln.mass_density(x_d+1,y_d) + Csln.mass_density(x_d+1,y_d+1) );

```

```

% Momentum balance at the anode solution - membrane interface
d_SV(Pointer.asln.y_vel(x_r,Grid_size.y_d_num)) = SCB.asln.mass_density.y(x_r,y_r+1)^-1 * ...
( ...
... % y-direction advection, migration and diffusion of y-direction momentum (out) - (in)
- ( abs( Asln.y_vel(x_d,Grid_size.y_d_num+1) ) ...
* Asln.mass_density(x_d,Grid_size.y_d_num+2) * Asln.y_vel(x_d,Grid_size.y_d_num+1) ...
- abs( VCB.y.asln.y_vel(x_r,Grid_size.y_d_num) ) ...
* Asln.mass_density(x_d,Grid_size.y_d_num+1) * VCB.y.asln.y_vel(x_r,Grid_size.y_d_num) ) ...
/ ( 0.5 * Geometry.y_d_size(Grid_size.y_d_num+1) ) ...
... % x-direction advection, migration and diffusion of y-direction momentum (out) - (in)
- ( Adv.y.asln.x_vel(x_r+1,y_r) * Adv.y.asln.y_vel(x_r+1,y_r) ...
* Adv.y.asln.mass_density(x_r+1,y_r) - Adv.y.asln.x_vel(x_r, y_r) ...
* Adv.y.asln.y_vel(x_r, y_r) * Adv.y.asln.mass_density(x_r, y_r) ) ...
/ Geometry.x_d_size(x_d) ...
... % y-direction normal forces due to pressure on the upwind and downwind faces
... (downwind) - (upwind)
- ( Asln.press(x_d,Grid_size.y_d_num+2) - Asln.press(x_d,Grid_size.y_d_num+1) ) ...
/ ( 0.5 * Geometry.y_d_size(Grid_size.y_d_num+1) ) ...
... % y-direction shear forces due to viscosity
+ Properties.anode.viscosity * ...
( d2_vy_d2_y_asln(x_r,Grid_size.y_d_num) + d2_vy_d2_x_asln(x_r,Grid_size.y_d_num) ) ...
... % Body force due to electric field acting on ions in the solution
- SCB.asln.charge_density.y(x_r,y_r+1) * ...
( Asln.elec_pot(x_d, y_d+1) - Asln.elec_pot(x_d,y_d) ) ...
/ ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d+1) ) ) ...
... % d rho / d t term from the LHS
- Asln.y_vel(x_d,Grid_size.y_d_num+1) * d_rho_dt.asln(x_r,Grid_size.y_d_num) ...
);

% Momentum balance at the membrane - cathode solution interface.
% (total membrane mass flux in)-(total advection mass flux out)
% -(migration + diffusion mass fluxes)=0
d_SV(Pointer.csln.y_vel(x_r,Grid_size.y_d_num)) = SCB.csln.mass_density.y(x_r,y_r+1)^-1 * ...
( ...
... % y-direction advection, migration and diffusion of y-direction momentum (out) - (in)
- ( abs( Csln.y_vel(x_d,Grid_size.y_d_num+1) ) ...

```

```

* Csln.mass_density(x_d,Grid_size.y_d_num+2) * Csln.y_vel(x_d,Grid_size.y_d_num+1) ...
- abs( VCB.y.csln.y_vel(x_r,Grid_size.y_d_num) ) ...
* Csln.mass_density(x_d,Grid_size.y_d_num+1) * VCB.y.csln.y_vel(x_r,Grid_size.y_d_num) )...
/ ( 0.5 * Geometry.y_d_size(Grid_size.y_d_num+1) ) ...
... % x-direction advection, migration and diffusion of y-direction momentum (out) - (in)
- ( Adv.y.csln.x_vel(x_r+1,y_r) * Adv.y.csln.y_vel(x_r+1,y_r) ...
* Adv.y.csln.mass_density(x_r+1,y_r) ...
- Adv.y.csln.x_vel(x_r, y_r) * Adv.y.csln.y_vel(x_r, y_r) ...
* Adv.y.csln.mass_density(x_r, y_r) ) / Geometry.x_d_size(x_d) ...
... % y-direction normal forces due to pressure on the upwind and downwind faces
... (downwind) - (upwind)
- ( Csln.press(x_d,Grid_size.y_d_num+2) - Csln.press(x_d,Grid_size.y_d_num+1) ) ....
/ ( 0.5 * Geometry.y_d_size(Grid_size.y_d_num+1) ) ...
... % y-direction shear forces due to viscosity
+ Properties.cathode.viscosity ...
* ( d2_vy_d2_y_csln(x_r,Grid_size.y_d_num) + d2_vy_d2_x_csln(x_r,Grid_size.y_d_num) ) ...
... % Body force due to electric field acting on ions in the solution
- SCB.csln.charge_density.y(x_r,y_r+1) ...
* ( Csln.elec_pot(x_d, y_d+1) - Csln.elec_pot(x_d,y_d) ) ...
/ ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d+1) ) ) ...
... % d rho / d t term from the LHS
- Csln.y_vel(x_d,Grid_size.y_d_num+1) * d_rho_dt.csln(x_r,Grid_size.y_d_num) ...
);

```

end

%% 3. SOLVE FOR MASS FRACTIONS USING SPECIES CONSERVATION EQUATIONS

for x_d = 2 : Grid_size.x_d_num + 1 % Run through all real cells between the inlet and outlet in the DCS

x_r = x_d - 1; % Corresponding position in the RCS

for y_d = 2 : Grid_size.y_d_num + 1

% Run through all real cells between the electrode and membrane in the DCS

```

y_r = y_d - 1; % Corresponding position in the RCS

% CALCULATE THE MASS FRACTION RATE OF CHANGE IN THE ELECTROLYTE SOLUTION
% (DETERMINE Y_k IN THE ELECTROLYTE SOLUTION)
% Loop through each species in each cell, enforcing species conservation.

for s_i = 1:Species.fuel.num

    d_SV(Pointer.asln.mass_fracs(x_r,y_r) + s_i - 1) = ...
    ... Divide by total mass in cell x_d,y_d to make the mass of species s_i in the cell into a MF
    ( Geometry.volume(x_d,y_d) * Asln.mass_density(x_d,y_d) )^-1 * ...
    ... Net advection of species s_i in the x-direction ( in - out )
    ( Geometry.x_flux_area(y_d) * ( SCB.asln.mass_density.x(x_r, y_r) ...
    * Asln.x_vel(x_r, y_d) * SCB.asln.Mass_fracs.x(x_r, y_r,s_i) ...
    - SCB.asln.mass_density.x(x_r+1,y_r) * Asln.x_vel(x_r+1,y_d) ...
    * SCB.asln.Mass_fracs.x(x_r+1,y_r,s_i) ) ...
    ... Net advection of species s_i in the y-direction ( in - out )
    + Geometry.y_flux_area(x_d) * ( SCB.asln.mass_density.y(x_r,y_r) ...
    * Asln.y_vel(x_d,y_r) * SCB.asln.Mass_fracs.y(x_r,y_r, s_i) ...
    - SCB.asln.mass_density.y(x_r,y_r+1) * Asln.y_vel(x_d,y_r+1) ...
    * SCB.asln.Mass_fracs.y(x_r,y_r+1,s_i) ) ...
    ... Net diffusion and migration of species s_i in the x-direction ( in - out )
    + Geometry.x_flux_area(y_d) ...
    * ( Xfluxes.mass.asln(x_r,y_r,s_i) - Xfluxes.mass.asln(x_r+1,y_r,s_i) ) ...
    ... Net diffusion and migration of species s_i in the y-direction ( in - out )
    + Geometry.y_flux_area(x_d) * ...
    ( Yfluxes.mass.asln(x_r,y_r,s_i) - Yfluxes.mass.asln(x_r,y_r+1,s_i) ) ...
    ... Change in the amount of species s_i due to change in density of s_i
    - Geometry.volume(x_d,y_d) * Asln.Mass_fracs(x_d,y_d,s_i) * d_rho_dt.asln(x_r,y_r) );

end

for s_i = 1:Species.oxidizer.num

    d_SV(Pointer.csln.mass_fracs(x_r,y_r) + s_i - 1) = ...

```

```

... Divide by total mass in cell x_d,y_d to make the mass of species s_i in the cell into a MF
( Geometry.volume(x_d,y_d) * Csln.mass_density(x_d,y_d) )^-1 * ...
... Net advection of species s_i in the x-direction ( in - out )
( Geometry.x_flux_area(y_d) * ( SCB.csln.mass_density.x(x_r, y_r) * Csln.x_vel(x_r, y_d) ...
  * SCB.csln.Mass_fractions.x(x_r, y_r,s_i) - SCB.csln.mass_density.x(x_r+1,y_r) ...
  * Csln.x_vel(x_r+1,y_d) * SCB.csln.Mass_fractions.x(x_r+1,y_r,s_i) ) ...
... Net advection of species s_i in the y-direction ( in - out )
+ Geometry.y_flux_area(x_d) * ( SCB.csln.mass_density.y(x_r,y_r) ...
  * Csln.y_vel(x_d,y_r) * SCB.csln.Mass_fractions.y(x_r,y_r, s_i) ...
  - SCB.csln.mass_density.y(x_r,y_r+1) * Csln.y_vel(x_d,y_r+1) ...
  * SCB.csln.Mass_fractions.y(x_r,y_r+1,s_i) ) ...
... Net diffusion and migration of species s_i in the x-direction ( in - out )
+ Geometry.x_flux_area(y_d) * ( Xfluxes.mass.csln(x_r,y_r,s_i) ...
  - Xfluxes.mass.csln(x_r+1,y_r,s_i) ) ...
... Net diffusion and migration of species s_i in the y-direction ( in - out )
+ Geometry.y_flux_area(x_d) * ( Yfluxes.mass.csln(x_r,y_r,s_i) ...
  - Yfluxes.mass.csln(x_r,y_r+1,s_i) ) ...
... Change in the amount of species s_i due to change in density of s_i
- Geometry.volume(x_d,y_d) * Csln.Mass_fractions(x_d,y_d,s_i) * d_rho_dt.csln(x_r,y_r) );

```

end

```

% Correct water mass fractions to ensure that all mass fractions sum to one for every cell.
d_SV(Pointer.asln.mass_fractions(x_r,y_r) + Pointer.anode.species.H2O - 1) = ....
  1 - sum( Asln.Mass_fractions(x_d,y_d,:) );
d_SV(Pointer.csln.mass_fractions(x_r,y_r) + Pointer.cathode.species.H2O - 1) = ...
  1 - sum( Csln.Mass_fractions(x_d,y_d,:) );

```

end

```

% ENFORCE SPECIES CONSERVATION AT THE ELECTRODE AND MEMBRANE INTERFACES
% (DETERMINE Y_k AT THE INTERFACES)

```

```

% Note that the equations in this section are operating on all species at once at each point along
% the channel channel by adding and subtracting column vectors.

```

```

% Anode - anode electrolyte interface
% 0 = (Anode reaction fluxes in) - (Electrolyte diffusion & migration fluxes out)
%   - (Convection fluxes out)
d_SV(Pointer.a_int.mass_fracs(x_r) + (1 : Species.fuel.num) - 1) = ...
  Yfluxes.mass.arxn(x_r,:) - squeeze(Yfluxes.mass.asln(x_r,1,:)) ...
  - A_int.Mass_fracs(x_r,:) * A_int.mass_density(x_r) * Asln.y_vel(x_d,1);

% Membrane - anode electrolyte interface
% 0 = (Electrolyte diffusion & migration mass fluxes of species k in) +
%      (Convection mass fluxes of species k in) - (Membrane mass fluxes of species k out)
d_SV(Pointer.m_int_a.mass_fracs(x_r) + (1 : Species.fuel.num) - 1) = ...
  squeeze(Yfluxes.mass.asln(x_r,Grid_size.y_d_num + 1,:)) ...
  + M_int_a.Mass_fracs(x_r,:) * M_int_a.mass_density(x_r) * Asln.y_vel(x_d,Grid_size.y_d_num + 1) ...
  - Yfluxes.mass.am(x_r,:);

% Membrane - cathode electrolyte interface
% 0 = (Electrolyte diffusion & migration mass fluxes of species k in) +
%      (Convection mass fluxes of species k in) - (Membrane mass fluxes of species k out)
d_SV(Pointer.m_int_c.mass_fracs(x_r) + (1 : Species.oxidizer.num) - 1) = ...
  squeeze(Yfluxes.mass.csln(x_r,Grid_size.y_d_num + 1,:)) ...
  + M_int_c.Mass_fracs(x_r,:) * M_int_c.mass_density(x_r) * Csln.y_vel(x_d,Grid_size.y_d_num + 1) ...
  - Yfluxes.mass.cm(x_r,:);

% Cathode - cathode electrolyte interface
% 0 = (Cathode reaction fluxes in) - (Electrolyte diffusion & migration fluxes out)
%   - (Convection fluxes out)
d_SV(Pointer.c_int.mass_fracs(x_r) + (1 : Species.oxidizer.num) - 1) = ...
  Yfluxes.mass.crxn(x_r,:) - squeeze(Yfluxes.mass.csln(x_r,1,:)) - C_int.Mass_fracs(x_r,:) ...
  * C_int.mass_density(x_r) * Csln.y_vel(x_d,1);

% Correct water mass fractions at the interfaces to ensure that all mass fractions sum to one.
d_SV(Pointer.a_int.mass_fracs(x_r) + Pointer.anode.species.H2O - 1) = ...
  1 - sum( A_int.Mass_fracs(x_r,:) );
d_SV(Pointer.m_int_a.mass_fracs(x_r) + Pointer.anode.species.H2O - 1) = ...

```

```

    1 - sum( M_int_a.Mass_fracs(x_r,:) );
d_SV(Pointer.m_int_c.mass_fracs(x_r) + Pointer.cathode.species.H2O - 1) = ...
    1 - sum( M_int_c.Mass_fracs(x_r,:) );
d_SV(Pointer.c_int.mass_fracs(x_r) + Pointer.cathode.species.H2O - 1) = ...
    1 - sum( C_int.Mass_fracs(x_r,:) );

```

```
end
```

```
%% 4. SOLVE FOR ELECTRIC POTENTIAL BY ENFORCING ELECTRONEUTRALITY
```

```

% Enforce electroneutrality: The solution can only be neutral if the species charges balance (no charge
% storage) and charge is conserved (charge fluxes in = charge fluxes out). Several state variables are
% involved but the dominant one is the electric potential, which is why these equations are used to solve
% for the electric potential.

```

```
% ENFORCE ELECTRONEUTRALITY AT THE ELECTRODE INTERFACES (SOLVE FOR ELECTRIC POTENTIAL)
```

```

% Calculate the electric potential just outside the anode and cathode double layers by forcing the
% solution to be electrically neutral.

```

```

d_SV(Pointer.a_int.elec_pot) = A_int.charge_density;
d_SV(Pointer.c_int.elec_pot) = C_int.charge_density;

```

```
% ENFORCE ELECTRONEUTRALITY AT THE MEMBRANE INTERFACES (SOLVE FOR ELECTRIC POTENTIAL)
```

```

% Calculate the electric potential just outside the double layers on the anode and cathode sides of
% the membrane by finding the values which produce an electrically neutral combinations of mass
% fractions.

```

```

d_SV(Pointer.m_int_a.elec_pot) = M_int_a.charge_density;
d_SV(Pointer.m_int_c.elec_pot) = M_int_c.charge_density;

```

```
% ENFORCE ELECTRONEUTRALITY IN THE BULK SOLUTION (SOLVE FOR ELECTRIC POTENTIAL)
```

```
if Flags.model.electroneutrality == 1
```

```

    for x_d = 2 : Grid_size.x_d_num + 1

```

```

        % Run through all real cells between the inlet and outlet in the DCS
    end
end

```

```

x_r = x_d - 1; % Corresponding position in the RCS
for y_d = 2 : Grid_size.y_d_num + 1
    % Run through all real cells between the electrode and membrane in the DCS
    y_r = y_d - 1; % Corresponding position in the RCS
    d_SV(Pointer.asln.elec_pot(x_r,y_r)) = Asln.charge_density(x_d,y_d);
    d_SV(Pointer.csln.elec_pot(x_r,y_r)) = Csln.charge_density(x_d,y_d);
end
end
else

for x_d = 2 : Grid_size.x_d_num + 1 % Run through all real cells between the inlet and outlet in DCS
x_r = x_d - 1; % Corresponding position in the RCS
for y_d = 2 : Grid_size.y_d_num + 1
    % Run through all real cells between the electrode and membrane in the DCS
    y_r = y_d - 1; % Corresponding position in the RCS

Asln_lapacian = ( ( Asln.elec_pot(x_d+1,y_d) - Asln.elec_pot(x_d,y_d) ) ...
    / ( 0.5 * ( Geometry.x_d_size(x_d+1) + Geometry.x_d_size(x_d) ) ) - ...
    ( Asln.elec_pot(x_d,y_d) - Asln.elec_pot(x_d-1,y_d) ) ...
    / ( 0.5 * ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d-1) ) ) ) / Geometry.x_d_size(x_d) ...
+ ( ( Asln.elec_pot(x_d,y_d+1) - Asln.elec_pot(x_d,y_d) ) ...
    / ( 0.5 * ( Geometry.y_d_size(y_d+1) + Geometry.y_d_size(y_d) ) ) - ...
    ( Asln.elec_pot(x_d,y_d) - Asln.elec_pot(x_d,y_d-1) ) ...
    / ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d-1) ) ) ) / Geometry.y_d_size(y_d);

Csln_lapacian = ( ( Csln.elec_pot(x_d+1,y_d) - Csln.elec_pot(x_d,y_d) ) ...
    / ( 0.5 * ( Geometry.x_d_size(x_d+1) + Geometry.x_d_size(x_d) ) ) - ...
    ( Csln.elec_pot(x_d,y_d) - Csln.elec_pot(x_d-1,y_d) ) ...
    / ( 0.5 * ( Geometry.x_d_size(x_d) + Geometry.x_d_size(x_d-1) ) ) ) / Geometry.x_d_size(x_d) ...
+ ( ( Csln.elec_pot(x_d,y_d+1) - Csln.elec_pot(x_d,y_d) ) ...
    / ( 0.5 * ( Geometry.y_d_size(y_d+1) + Geometry.y_d_size(y_d) ) ) - ...
    ( Csln.elec_pot(x_d,y_d) - Csln.elec_pot(x_d,y_d-1) ) ...
    / ( 0.5 * ( Geometry.y_d_size(y_d) + Geometry.y_d_size(y_d-1) ) ) ) / Geometry.y_d_size(y_d);

d_SV(Pointer.asln.elec_pot(x_r,y_r)) = Asln_lapacian + Asln.charge_density(x_d,y_d)...

```

```

        / (Constants.permittivity * Constants.H2O_permeability);
    d_SV(Pointer.csln.elec_pot(x_r,y_r)) = Csln_lapacian + Csln.charge_density(x_d,y_d)...
        / (Constants.permittivity * Constants.H2O_permeability);
end

end

end

end

function [ SV_initial ] = FUNC_ADJUST_SOLN_VOLTAGE( SV_steady_state, V_cell_previous, V_cell_new, ...
    Scales, Pointer, Geometry, Grid_size )

%% SUMMARY: FUNC_ADJUST_SOLN_VOLTAGE
% Purpose: This function accepts a model solution and adjusts the electric potential at the interfaces
% and in the bulk to make it a reasonable initial guess for a different point on the polarization curve
% for the same set of operating conditions.
% Author: Rick Stroman

%% PRELIMINARY CALCS
% Determine the change in cell voltage to be accomodated
delta_V = V_cell_new - V_cell_previous;

% Find the unscaled values from the this solution
SV_steady_state_unsc = SV_steady_state .* Scales.SV;

%% ADJUST ELECTRIC POTENTIAL AT THE INTERFACES, MOVING EACH BY 20% OF THE TOTAL POTENTIAL CHANGE

SV_steady_state_unsc(Pointer.c_int.elec_pot) = ...
    SV_steady_state_unsc(Pointer.c_int.elec_pot) + 0.8 * delta_V;
SV_steady_state_unsc(Pointer.m_int_c.elec_pot) = ...

```

```

        SV_steady_state_unsc(Pointer.m_int_c.elec_pot) + 0.6 * delta_V;
SV_steady_state_unsc(Pointer.m_int_a.elec_pot) = ...
        SV_steady_state_unsc(Pointer.m_int_a.elec_pot) + 0.4 * delta_V;
SV_steady_state_unsc(Pointer.a_int.elec_pot)   = ...
        SV_steady_state_unsc(Pointer.a_int.elec_pot)   + 0.2 * delta_V;

%% ADJUST ELECTRIC POTENTIAL IN BULK ELECTROLYTE IN SO IT STILL MATCHES THE INTERFACES AT EACH SIDE

% Rates of change d(phi)/d(y) moving from each electrode toward the membrane
anode_potential_slope = (0.4 * delta_V - 0.2 * delta_V) / Geometry.channel_height;
cathode_potential_slope = (0.6 * delta_V - 0.8 * delta_V) / Geometry.channel_height;

% Figure out how much to adjust the electric potential in each bulk discretization, assuming
% that a given y-position at all x-positions is always shifted the same amount.
for y_d = 1:Grid_size.y_d_num

    asln_correction(y_d,1) = 0.2 * delta_V + anode_potential_slope * Geometry.y_d_location(y_d+1);
    csln_correction(y_d,1) = 0.8 * delta_V + cathode_potential_slope * Geometry.y_d_location(y_d+1);

end

% Apply the electric potential adjustment to the bulk cells
for x_d = 1 : Grid_size.x_d_num
    SV_steady_state_unsc(Pointer.asln.elec_pot(x_d,:)) = ...
        SV_steady_state_unsc(Pointer.asln.elec_pot(x_d,:)) + asln_correction;
    SV_steady_state_unsc(Pointer.csln.elec_pot(x_d,:)) = ...
        SV_steady_state_unsc(Pointer.csln.elec_pot(x_d,:)) + csln_correction;
end

% Return the adjusted solution vector for use as the initial guess at the next point
SV_initial = SV_steady_state_unsc ./ Scales.SV;

end

```

7.2 DBFC Model Calibration Code

```
%% SUMMARY: DBFC_CALIBRATE
% Purpose: Try varying model parameters to match experimental data (pol curve)
% Author: Rick Stroman

%% SET UP THE WORKSPACE

clear all; close all; clc;

global calls param_scales measured_current_densities

tic; calls = 0;

%% INITIALIZE VARIABLES, GUESSES AND SCALES FOR MODEL PARAMETERS

% Anode reaction #1
anode_rxn1_kf = 0.0070582;  anode_rxn1_betaf = 0.1043;

% Anode reaction #3
anode_rxn3_kf = 0.00016476;

% Cathode reaction #1
cathode_rxn1_kr = 0.0077536;  cathode_rxn1_betar = 0.47467;

% Cathode reaction #3
cathode_rxn3_kf = 0.00059533;

% Cathode reaction #4
cathode_rxn4_kr = 2.4938e-09;  cathode_rxn4_betar = 0.33742;

% Cathode reaction #2
cathode_rxn2_kr = 1e-2;  cathode_rxn2_betar = 0.5;
```

```

% Electrode roughness factors
anode_roughness = 2.5784; cathode_roughness = 4.1093;

param_scales = [ 1e-3; 1e-1; 1e-4; 1e-3; 1e-1; 1e-4; 1e-9; 1e-1; 1e-2; 1e-1; ...
1e0; 1e0];

%% ESTABLISH BOUNDS FOR FITTED PARAMETERS

% Lower bounds on fitted parameters.
lsqnonlin_lower_bounds = [ ...
1e-15; 1e-3; 1e-15; 1e-15; 1e-3; 1e-15; 1e-20; 1e-3; 1e-15; 1e-3;
1; % Anode roughness factor
1]; % Cathode roughness factor

lsqnonlin_lower_bounds = lsqnonlin_lower_bounds ./ param_scales ;

% Upper bounds on fitted parameters.
lsqnonlin_upper_bounds = [
1e2; 1e0; 1e2; 1e2; 1e0; 1e2; 1e-1; 1e0; 1e1; 1e0;
30; % Anode roughness factor
30]; % Cathode roughness factor

lsqnonlin_upper_bounds = lsqnonlin_upper_bounds ./ param_scales ;

%% SET UP THE INITIAL GUESS

Parameters_guess = [

anode_rxn1_kf anode_rxn1_betaf anode_rxn3_kf cathode_rxn1_kr cathode_rxn1_betar cathode_rxn3_kf ...
cathode_rxn4_kr cathode_rxn4_betar cathode_rxn2_kr cathode_rxn2_betar anode_roughness cathode_roughness];

%% CALL THE SOLVER TO FIT THE MODE TO THE MEASUREMENTS

lsqnonlin_options = optimset('Algorithm', 'trust-region-reflective', 'Display', 'final', 'PlotFcns', ...

```

```

@optimplotresnorm );

[Parameters_fitted, residual_2norm, residuals] = ...
    lsqnonlin(@DBFC_FIT_ERROR, Parameters_guess ./ param_scales, lsqnonlin_lower_bounds, ...
    lsqnonlin_upper_bounds, lsqnonlin_options);

%% DISPLAY THE RESULTS

Parameters_fitted_unsc = param_scales .* Parameters_fitted;

% Fitted parameters
disp('-----')
disp('The fitted parameters are: ')
disp(['Anode forward rate constant for reaction #1:      ' num2str(Parameters_fitted_unsc(1))])
disp(['Anode forward symmetry factor for reaction #1:    ' num2str(Parameters_fitted_unsc(2))])
disp(['Anode forward rate constant for reaction #3:      ' num2str(Parameters_fitted_unsc(3))])
disp(['Cathode reverse rate constant for reaction #1:    ' num2str(Parameters_fitted_unsc(4))])
disp(['Cathode reverse symmetry factor for reaction #1:  ' num2str(Parameters_fitted_unsc(5))])
disp(['Cathode forward rate constant for reaction #3:    ' num2str(Parameters_fitted_unsc(6))])
disp(['Cathode reverse rate constant for reaction #4:    ' num2str(Parameters_fitted_unsc(7))])
disp(['Cathode reverse symmetry factor for reaction #4:  ' num2str(Parameters_fitted_unsc(8))])
disp(['Cathode reverse rate constant for reaction #2:    ' num2str(Parameters_fitted_unsc(9))])
disp(['Cathode reverse symmetry factor for reaction #2:  ' num2str(Parameters_fitted_unsc(10))])
disp(['Anode roughness factor:      ' num2str(Parameters_fitted_unsc(11))])
disp(['Cathode roughness factor:     ' num2str(Parameters_fitted_unsc(12))])

% Display the results

SStotal = (length(measured_current_densities)-1) * var(measured_current_densities);
SSresid = sum(residuals.^2);
R_squared = 1-SSresid/SStotal;

disp(' ')

```

```

disp(['The best fit resulted in an error 2-norm of: ' num2str(norm(residuals))])
disp(['The R^2 value for the best fit was: ' num2str(R_squared)])
disp('-----')

function [Errors] = DBFC_FIT_ERROR(Parameters_guess)

%% SUMMARY: DBFC_FIT_ERROR
% Purpose: Run the DBFC model to estimate the difference between calculated and
% measured current density, at a series of cell voltages.`
% Author: Rick Stroman

%% 1. SET UP THE PARAMETERS AND SCENARIO

global param_scales Flags Errors_recent calls BC total_cathode_current Scales Geometry Pointer

calls = calls + 1;

Parameters_guess_unsc = param_scales .* Parameters_guess;
anode_rxn1_kf          = Parameters_guess_unsc(1);
anode_rxn1_betaf      = Parameters_guess_unsc(2);
anode_rxn3_kf         = Parameters_guess_unsc(3);
cathode_rxn1_kr       = Parameters_guess_unsc(4);
cathode_rxn1_betar    = Parameters_guess_unsc(5);
cathode_rxn3_kf       = Parameters_guess_unsc(6);
cathode_rxn4_kr       = Parameters_guess_unsc(7);
cathode_rxn4_betar    = Parameters_guess_unsc(8);
cathode_rxn2_kr       = Parameters_guess_unsc(9);
cathode_rxn2_betar    = Parameters_guess_unsc(10);
anode_roughness       = Parameters_guess_unsc(11);
cathode_roughness     = Parameters_guess_unsc(12);

disp(' ')
disp('-----')

```

```

disp(['FUNCTION CALL NUMBER: ' num2str(calls)])
disp('-----')
disp(['Time since fitting algorithm began: ' num2str(toc/60) ' min'])
disp('Parameters for the fit:')
disp(['Anode forward rate constant for reaction #1:      ' num2str(Parameters_guess_unsc(1))])
disp(['Anode forward symmetry factor for reaction #1:    ' num2str(Parameters_guess_unsc(2))])
disp(['Anode forward rate constant for reaction #3:      ' num2str(Parameters_guess_unsc(3))])
disp(['Cathode reverse rate constant for reaction #1:    ' num2str(Parameters_guess_unsc(4))])
disp(['Cathode reverse symmetry factor for reaction #1:  ' num2str(Parameters_guess_unsc(5))])
disp(['Cathode forward rate constant for reaction #3:    ' num2str(Parameters_guess_unsc(6))])
disp(['Cathode reverse rate constant for reaction #4:    ' num2str(Parameters_guess_unsc(7))])
disp(['Cathode reverse symmetry factor for reaction #4:  ' num2str(Parameters_guess_unsc(8))])
disp(['Cathode reverse rate constant for reaction #2:    ' num2str(Parameters_guess_unsc(9))])
disp(['Cathode reverse symmetry factor for reaction #2:  ' num2str(Parameters_guess_unsc(10))])
disp(['Anode roughness factor:      ' num2str(Parameters_guess_unsc(11))])
disp(['Cathode roughness factor:     ' num2str(Parameters_guess_unsc(12))])
disp('Solving the next polarization curve...')
disp('-----')
disp(' ')

%% 2. SET OTHER PARAMETERS NEEDED FOR THE FIT

num_curves = 3;

BH4_conc = [ 2.5e-3; 10e-3; 20e-3]; % M
% H2O2_conc = [ 40e-3; 40e-3; 40e-3; 40e-3 ]; % M
% fuel_flow_rate = 1.6666e-7; % m^3/s
% oxidizer_flow_rate = 1.6666e-7; % m^3/s

% Exp54B
Voltages{1,:} = [ 1.6156 1.525 1.5 1.475 1.45 1.4 1.3 1.2 1.1 1.0 0.9 0.8 0.7 0.6 0.5 ];
% Exp54G
Voltages{2,:} = [ 1.629 1.525 1.5 1.475 1.45 1.4 1.35 1.3 1.2 1.1 1.0 0.9 0.8 0.7 0.6 0.5 0.4 0.3 ];
% Exp54E
Voltages{3,:} = [ 1.61 1.525 1.5 1.475 1.45 1.4 1.35 1.3 1.2 1.1 1.0 0.9 0.8 0.7 0.6 0.5 0.4 ];

```

```

Measured_currents{:,1} = 2.5*[ 0    0.465 0.617 0.787 0.964 1.353 3.064 4.092 4.940 5.620 6.286 ...
    6.805 7.118 7.286 7.455]; % Exp54B
Measured_currents{:,2} = 2.5*[ 0.00 0.72  1.00  1.67  2.06  3.53  5.07  7.14  10.27 12.61 14.22 ...
    16.25 17.84 19.29 20.74 21.80 22.51 23.54 ]; % Exp54G in mA/cm^2
Measured_currents{:,3} = 2.5*[ 0.00 0.59  0.87  1.53  2.31  4.19  6.46  9.76  16.04 19.62 21.42 ...
    22.87 24.06 25.27 26.51 28.05 29.57 ]; % Exp 54E

Files{1} = 'Inter_Results_2013-10-4-12h-14m-55.621s_1.6156V.mat';
Files{2} = 'Inter_Results_2013-10-3-11h-24m-40.958s_1.629V.mat';
Files{3} = 'Exp54E_workspace.mat';

error_scales = ones(50,1); error_scales(10:15) = 1e0;
error_scales(25:33) = 1e0; error_scales(43:50) = 1e0;

%% 3. SOLVE THE MODEL

Calculated_current_densities = []; Measured_current_densities = [];

for curve_index = 1:1:num_curves

    disp(' ')
    disp(['Solving the DBFC model for curve number: ' num2str(curve_index) '...']);    disp(' ')

    BH4_conc_loop = BH4_conc(curve_index);
    %     H2O2_conc_loop = H2O2_conc(curve_index);
    Voltages_loop = Voltages{curve_index,:};
    Seed_file = Files{curve_index};

    %% 2. SET UP THE MODEL

    DBFC_USER_INPUT;
    DBFC_CONSTANTS_AND_PROPERTIES;
    DBFC_PROCESS_USER_INPUT

```

```

DBFC_SETUP_REACTION_RATES
DBFC_INITIALIZE

Measured_current_density_loop = Measured_currents(:,curve_index) ...
    / (1000*Geometry.channel_length*Geometry.channel_width);

% Solve the model for each cell voltage specified in Cathode_electric_potential
for voltage_index = 1:length(Voltages_loop);

    % SET THE CELL VOLTAGE AND CALL THE REQUESTED SOLVER TO SOLVE THE MODEL

    BC.cathode.elec_pot = Voltages_loop(voltage_index);

    disp('-----')
    disp(['Solving point number: ' num2str(voltage_index) '...'])
    DBFC_KINSOL

    % Store the current density from this voltage in an array to return from this function
    Calculated_current_density_loop(voltage_index,1) = -total_cathode_current ...
        / (Geometry.channel_length*Geometry.channel_width);

    % Display status in the command window so we know what is going on
    disp(['Point number ', num2str(voltage_index), ' of ', num2str(length(Voltages_loop)), ...
        ' is complete.'])
    disp(['Cell voltage is: ' num2str(BC.cathode.elec_pot), ' V'])
    disp(['Total cell current is: ' num2str(total_cathode_current) ' A'])
    disp(['Average cell current density is: ' ...
        num2str(Calculated_current_density_loop(voltage_index)) ' A/m^2'])
    disp(['Time since start of calibration: ' num2str(toc/60) ' min']);
    disp('-----')

    if Flags.setup.reuse_previous_soln && voltage_index < length(Voltages_loop)

        % Configure the solution from the previous voltage to be the initial guess for the next voltage

```

```

if Flags.setup.adjust_V_prev_soln
    % Adjust the solution from the previous voltage to form the initial guess for the next point
    V_cell_old = Cathode_electric_potential(voltage_index);
    V_cell_new = Cathode_electric_potential(voltage_index+1);
    SV_initial = FUNC_ADJUST_SOLN_VOLTAGE( SV_steady_state, V_cell_old, V_cell_new, ...
        Scales, Pointer, Geometry );
else
    % Use the solution from the previous voltage as-is
    SV_initial = SV_steady_state;
end

end

end % Loop to walk down the polarization curve, one cell voltage at a time

% Plot error between current density predicted by the present set of fitted
% parameters and the literature values
figure(curve_index);
hold on;
plot(Measured_current_density_loop, Voltages_loop , 'b-s', Calculated_current_density_loop, ...
    Voltages_loop, 'r-o')
ylabel('Cell Voltage [V]')
xlabel('Current Density [A/m^2]')
legend('Measured', 'Calculated')
title('Fit Results')
hold off;

Calculated_current_densities = vertcat(Calculated_current_densities, Calculated_current_density_loop);
Measured_current_densities   = vertcat(Measured_current_densities, Measured_current_density_loop');

clear Voltages_loop Measured_current_density_loop Calculated_current_density_loop

end

%% PREPARE THE ERRORS FOR OUTPUT

```

```

Errors = error_scales .* (Calculated_current_densities - Measured_current_densities);
Errors_recent = Errors;

disp(' ')
disp(['2-norm of fit residuals is: ' num2str(norm(Errors_recent))])
disp(' ')

if Flags.setup.save_final_output
    c = clock;
    filename_identifier = strcat(num2str(c(1)), '-', num2str(c(2)), '-', num2str(c(3)), '-', ...
        num2str(c(4)), 'h-', num2str(c(5)), 'm-', num2str(c(6)), 's');
    filename = strcat('Results_', filename_identifier, '.mat');
    save(filename); disp(' '); disp(['Results were saved in file: ' filename]); disp(' ');
end

end

```

Chapter 8: Bibliography

1. Urian, R.C., et al., *Direct Borohydride/Hydrogen Peroxide Fuel Cell Development*. Proceedings of the 43rd Power Sources Conference, 2008. **43**: p. 295-298.
2. Urian, R.C., *Air Independent Fuel Cells Utilizing Borohydride and Hydrogen Peroxide*. Material Research Society Symposium Proceedings, 2010. **1213**.
3. Jacobson, M.M.K.a.R.W., *Ventrol Alembic*, 1979. **15**(2).
4. Bard, A.J. and L.R. Faulkner, *Electrochemical methods : fundamentals and applications*. 2nd ed. 2001, New York: Wiley. xxi, 833 p.
5. Kee, R.J., H.Y. Zhu, and D.G. Goodwin, *Solid-oxide fuel cells with hydrocarbon fuels*. Proceedings of the Combustion Institute, 2005. **30**: p. 2379-2404.
6. Bird, R.B., W.E. Stewart, and E.N. Lightfoot, *Transport phenomena*. 2nd, Wiley international ed. 2002, New York: J. Wiley. xii, 895 p.
7. de Leon, C.P., et al., *Direct borohydride fuel cells*. Journal of Power Sources, 2006. **155**(2): p. 172-181.
8. Amendola, S.C., et al., *A novel high power density borohydride-air cell*. Journal of Power Sources, 1999. **84**(1): p. 130-133.
9. Miley, G.H., et al., *Direct NaBH₄/H₂O₂ fuel cells*. Journal of Power Sources, 2007. **165**(2): p. 509-516.
10. Townea, S., et al., *Performance of a Direct Borohydride Fuel Cell*. ECS Transactions, 2009. **25**(1): p. 1951-1957.
11. Cheng, H. and K. Scott, *Influence of operation conditions on direct borohydride fuel cell performance*. Journal of Power Sources, 2006. **160**(1): p. 407-412.
12. Jamard, R., et al., *Study of fuel efficiency in a direct borohydride fuel cell*. Journal of Power Sources, 2008. **176**(1): p. 287-292.
13. Wu, H.J., et al., *Influence of operation conditions on direct NaBH(4)/H(2)O(2) fuel cell performance*. International Journal of Hydrogen Energy, 2010. **35**(7): p. 2648-2651.
14. de Leon, C.P., et al., *A direct borohydride-peroxide fuel cell using a Pd/Ir alloy coated microfibrinous carbon cathode*. Electrochemistry Communications, 2008. **10**(10): p. 1610-1613.

15. Urian, R.C., *Expanded 3D Electrode Architecture for Low Temperature Liquid Fuel Cells*. Materials Research Society Symposium Proceedings, 2009. **1168**.
16. de Leon, C.P., et al., *A direct borohydride - Acid peroxide fuel cell*. Journal of Power Sources, 2007. **164**(2): p. 441-448.
17. Raman, R.K., N.A. Choudhury, and A.K. Shukla, *A high output voltage direct borohydride fuel cell*. Electrochemical and Solid State Letters, 2004. **7**(12): p. A488-A491.
18. Li, Z.P., et al., *A fuel cell development for using borohydrides as the fuel*. Journal of the Electrochemical Society, 2003. **150**(7): p. A868-A872.
19. Cheng, H., et al., *Evaluation of new ion exchange membranes for direct borohydride fuel cells*. Journal of Membrane Science, 2007. **288**(1-2): p. 168-174.
20. Ma, J., N.A. Choudhury, and Y. Sahai, *A comprehensive review of direct borohydride fuel cells*. Renewable & Sustainable Energy Reviews, 2010. **14**(1): p. 183-199.
21. Cao, D.X., et al., *Kinetics of hydrogen peroxide electroreduction on Pd nanoparticles in acidic medium*. Journal of Electroanalytical Chemistry, 2008. **621**(1): p. 31-37.
22. Choudhary, V.R., C. Samanta, and T.V. Choudhary, *Factors influencing decomposition of H₂O₂ over supported Pd catalyst in aqueous medium*. Journal of Molecular Catalysis a-Chemical, 2006. **260**(1-2): p. 115-120.
23. Choudhury, N.A., et al., *An alkaline direct borohydride fuel cell with hydrogen peroxide as oxidant*. Journal of Power Sources, 2005. **143**(1-2): p. 1-8.
24. Demirci, U.B., *Direct borohydride fuel cell: Main issues met by the membrane-electrodes-assembly and potential solutions*. Journal of Power Sources, 2007. **172**(2): p. 676-687.
25. Merino-Jimenez, I., et al., *Developments in direct borohydride fuel cells and remaining challenges*. Journal of Power Sources, 2012. **219**: p. 339-357.
26. Santos, D.M.F. and C.A.C. Sequeira, *On the electrosynthesis of sodium borohydride*. International Journal of Hydrogen Energy, 2010. **35**(18): p. 9851-9861.
27. Santos, D.M.F. and C.A.C. Sequeira, *Sodium borohydride as a fuel for the future*. Renewable & Sustainable Energy Reviews, 2011. **15**(8): p. 3980-4001.
28. Luo, N., et al., *NaBH₄/H₂O₂ fuel cells for air independent power systems*. Journal of Power Sources, 2008. **185**(2): p. 685-690.

29. Lakernan, J.B., et al., *The direct borohydride fuel cell for UUV propulsion power*. Journal of Power Sources, 2006. **162**(2): p. 765-772.
30. Ahmed, M. and I. Dincer, *A review on methanol crossover in direct methanol fuel cells: challenges and achievements*. International Journal of Energy Research, 2011. **35**(14): p. 1213-1228.
31. Li, X.L. and A. Faghri, *Review and advances of direct methanol fuel cells (DMFCs) part I: Design, fabrication, and testing with high concentration methanol solutions*. Journal of Power Sources, 2013. **226**: p. 223-240.
32. Bahrami, H. and A. Faghri, *Review and advances of direct methanol fuel cells: Part II: Modeling and numerical simulation*. Journal of Power Sources, 2013. **230**: p. 303-320.
33. Retnamma, R., A.Q. Novais, and C.M. Rangel, *Kinetics of hydrolysis of sodium borohydride for hydrogen production in fuel cell applications: A review*. International Journal of Hydrogen Energy, 2011. **36**(16): p. 9772-9790.
34. Liu, B.H. and Z.P. Li, *Current status and progress of direct borohydride fuel cell technology development*. Journal of Power Sources, 2009. **187**(2): p. 291-297.
35. Raman, R.K. and A.K. Shukla, *A direct borohydride/hydrogen peroxide fuel cell with reduced alkali crossover*. Fuel Cells, 2007. **7**(3): p. 225-231.
36. Rostamikia, G., et al., *First-principles based microkinetic modeling of borohydride oxidation on a Au(111) electrode*. Journal of Power Sources, 2011. **196**(22): p. 9228-9237.
37. Rostamikia, G. and M.J. Janik, *Borohydride Oxidation over Au(111): A First-Principles Mechanistic Study Relevant to Direct Borohydride Fuel Cells*. Journal of the Electrochemical Society, 2009. **156**(1): p. B86-B92.
38. Chatenet, M., M.B. Molina-Concha, and J.P. Diard, *First insights into the borohydride oxidation reaction mechanism on gold by electrochemical impedance spectroscopy*. Electrochimica Acta, 2009. **54**(6): p. 1687-1693.
39. Finkelstein, D.A., et al., *Rotating Disk Electrode (RDE) Investigation of BH₄⁻ and BH₃OH⁻ Electro-oxidation at Pt and Au: Implications for BH₄⁻ Fuel Cells*. Journal of Physical Chemistry C, 2009. **113**(45): p. 19700-19712.
40. Santos, D.M.F. and C.A.C. Sequeira, *Chronopotentiometric study of the electrooxidation of borohydride anion in alkaline medium*. Diffusion in Solids and Liquids, 2006. **258-260**: p. 333-339.
41. Chatenet, M., F.H.B. Lima, and E.A. Ticianelli, *Gold is not a Faradaic-Efficient Borohydride Oxidation Electrocatalyst: An Online Electrochemical Mass*

- Spectrometry Study*. Journal of the Electrochemical Society, 2010. **157**(5): p. B697-B704.
42. Freitas, K.S., et al., *Mass transport effects in the borohydride oxidation reaction- Influence of the residence time on the reaction onset and faradaic efficiency*. Catalysis Today, 2011. **170**(1): p. 110-119.
 43. Concha, B.M., et al., *In situ infrared (FTIR) study of the borohydride oxidation reaction*. Electrochemistry Communications, 2009. **11**(1): p. 223-226.
 44. Concha, B.M., et al., *In Situ Infrared (FTIR) Study of the Mechanism of the Borohydride Oxidation Reaction on Smooth Pt Electrode*. Journal of Physical Chemistry C, 2011. **115**(25): p. 12439-12447.
 45. Elder, J.P. and A. Hickling, *Anodic Behaviour of Borohydride Ion*. Transactions of the Faraday Society, 1962. **58**(477): p. 1852-&.
 46. Gardiner, J.A. and J.W. Collat, *Kinetics of Stepwise Hydrolysis of Tetrahydroborate Ion*. Journal of the American Chemical Society, 1965. **87**(8): p. 1692-&.
 47. Gardiner, J.A. and J.W. Collat, *Polarography Fo Tetrahydroborate Ion . Effect of Hydrolysis on System*. Inorganic Chemistry, 1965. **4**(8): p. 1208-&.
 48. Mirkin, M.V., H.J. Yang, and A.J. Bard, *Borohydride Oxidation at a Gold Electrode*. Journal of the Electrochemical Society, 1992. **139**(8): p. 2212-2217.
 49. Chatenet, M., et al., *Kinetics of sodium borohydride direct oxidation and oxygen reduction in sodium hydroxide electrolyte - Part I. BH₄⁻ electro-oxidation on Au and Ag catalysts*. Electrochimica Acta, 2006. **51**(25): p. 5459-5467.
 50. Krishnan, P., et al., *Rotating ring-disc electrode (RRDE) investigation of borohydride electro-oxidation*. Journal of Power Sources, 2008. **182**(1): p. 106-111.
 51. Concha, B.M., et al., *In situ infrared (FTIR) study of the mechanism of the borohydride oxidation reaction*. Physical Chemistry Chemical Physics, 2010. **12**(37): p. 11507-11516.
 52. Rostamikia, G. and M.J. Janik, *Direct borohydride oxidation: mechanism determination and design of alloy catalysts guided by density functional theory*. Energy & Environmental Science, 2010. **3**(9): p. 1262-1274.
 53. Cheng, H. and K. Scott, *Determination of kinetic parameters for borohydride oxidation on a rotating Au disk electrode*. Electrochimica Acta, 2006. **51**(17): p. 3429-3433.

54. Liu, B.H., J.Q. Yang, and Z.P. Li, *Concentration ratio of [OH-]/[BH4-]: A controlling factor for the fuel efficiency of borohydride electro-oxidation*. International Journal of Hydrogen Energy, 2009. **34**(23): p. 9436-9443.
55. Flatgen, G., et al., *Autocatalytic mechanism of H₂O₂ reduction on Ag electrodes in acidic electrolyte: experiments and simulations*. Electrochimica Acta, 1999. **44**(25): p. 4499-4506.
56. Doblhofer, K., et al., *Autocatalysis by the intermediate surface hydroxide formed during hydrogen peroxide reduction on silver electrodes*. Surface Science, 2009. **603**(10-12): p. 1900-1903.
57. Adams, B.D., C.K. Ostrom, and A.C. Chen, *Highly Active PdPt Catalysts for the Electrochemical Reduction of H₂O₂*. Journal of the Electrochemical Society, 2011. **158**(4): p. B434-B439.
58. Bessette, R.R., et al., *A study of cathode catalysis for the aluminium hydrogen peroxide semi-fuel cell*. Journal of Power Sources, 1999. **80**(1-2): p. 248-253.
59. Pourbaix, M., *Atlas of electrochemical equilibria in aqueous solutions*. 2d English ed. 1974, Houston, Tex.: National Association of Corrosion Engineers. 644 p.
60. Santos, D.M.F. and C.A.C. Sequeira, *Determination of Kinetic and Diffusional Parameters for Sodium Borohydride Oxidation on Gold Electrodes*. Journal of the Electrochemical Society, 2009. **156**(5): p. F67-F74.
61. Santos, D.M.F. and C.A.C. Sequeira, *Chronopotentiometric Investigation of Borohydride Oxidation at a Gold Electrode*. Journal of the Electrochemical Society, 2010. **157**(1): p. F16-F21.
62. Finkelstein, D.A., et al., *Alternative Oxidants for High-Power Fuel Cells Studied by Rotating Disk Electrode (RDE) Voltammetry at Pt, Au, and Glassy Carbon Electrodes*. Journal of Physical Chemistry C, 2011. **115**(13): p. 6073-6084.
63. Weber, A.Z. and J. Newman, *Transport in polymer-electrolyte membranes - I. Physical model*. Journal of the Electrochemical Society, 2003. **150**(7): p. A1008-A1015.
64. Okada, T., et al., *Ion and water transport characteristics of perfluorosulfonated ionomer membranes with H⁺ and alkali metal cations*. Journal of Physical Chemistry B, 2002. **106**(6): p. 1267-1273.
65. Mauritz, K.A. and R.B. Moore, *State of understanding of Nafion*. Chemical Reviews, 2004. **104**(10): p. 4535-4585.
66. Liu, B.H. and S. Suda, *Influences of fuel crossover on cathode performance in a micro borohydride fuel cell*. Journal of Power Sources, 2007. **164**(1): p. 100-104.

67. Okada, T., et al., *Transport and equilibrium properties of Nafion (R) membranes with H⁺ and Na⁺ ions*. Journal of Electroanalytical Chemistry, 1998. **442**(1-2): p. 137-145.
68. Verma, A. and S. Basu, *Experimental evaluation and mathematical modeling of a direct alkaline fuel cell*. Journal of Power Sources, 2007. **168**(1): p. 200-210.
69. Sanli, A.E., M.L. Aksu, and B.Z. Uysal, *Advanced mathematical model for the passive direct borohydride/peroxide fuel cell*. International Journal of Hydrogen Energy, 2011. **36**(14): p. 8542-8549.
70. Shah, A.A., et al., *Mathematical modelling of direct borohydride fuel cells*. Journal of Power Sources, 2013. **221**: p. 157-171.
71. Byrd, E.D. and G.H. Miley, *Simulation studies of the membrane exchange assembly of an all-liquid, proton exchange membrane fuel cell*. Journal of Power Sources, 2008. **176**(1): p. 222-228.
72. Sprague, I.B. and P. Dutta, *Modeling of Diffuse Charge Effects in a Microfluidic Based Laminar Flow Fuel Cell*. Numerical Heat Transfer Part a-Applications, 2011. **59**(1): p. 1-27.
73. Sprague, I. and P. Dutta, *Role of the diffuse layer in acidic and alkaline fuel cells*. Electrochimica Acta, 2011. **56**(12): p. 4518-4525.
74. Ottonello, G., *Principles of geochemistry*. 1997, New York: Columbia University Press. xii, 894 p.
75. Schumb, W.C., *Hydrogen peroxide*. American Chemical Society Monograph series,. 1955, New York,: Reinhold Pub. Corp. xiii, 759 p.
76. Oran, E.S. and J.P. Boris, *Numerical simulation of reactive flow*. 2nd ed. 2001, Cambridge, U.K. ; New York, NY: Cambridge University Press. xix, 529 p.
77. Newman, J.S. and K.E. Thomas-Alyea, *Electrochemical systems*. 3rd ed. 2004, Hoboken, N.J.: J. Wiley. xx, 647 p.
78. Cloutier, C.R., A. Alfantazi, and E. Gyenge, *Physicochemical Transport Properties of Aqueous Sodium Metaborate Solutions for Sodium Borohydride Hydrogen Generation and Storage and Fuel Cell Applications*. Thermec 2006 Supplement, 2007. **15-17**: p. 267-274.
79. Newman, J., *Current Distribution on a Rotating Disk Below Limiting Current*. Journal of the Electrochemical Society, 1966. **113**(12): p. 1235-&.
80. Nielsen, J.M., A.W. Adamson, and J.W. Cobble, *The Self-Diffusion Coefficients of the Ions in Aqueous Sodium Chloride and Sodium Sulfate at 25-Degrees*. Journal of the American Chemical Society, 1952. **74**(2): p. 446-451.

81. Poisson, A. and J. Chanu, *Semi-Empirical Equations for the Partial Molar Volumes of Some Ions in Water and Seawater*. Mar. Chem., 1980. **8**: p. 289-298.
82. Lakshminarayanaiah, N., *Transport phenomena in membranes*. 1969, New York,: Academic Press. xi, 517 p.
83. Santos, D.M.F. and C.A.C. Sequeira, *Effect of Membrane Separators on the Performance of Direct Borohydride Fuel Cells*. Journal of the Electrochemical Society, 2012. **159**(2): p. B126-B132.
84. Evans, C.E., et al., *Role of conditioning on water uptake and hydraulic permeability of Nafion (R) membranes*. Journal of Membrane Science, 2006. **279**(1-2): p. 521-528.
85. Gu, L.F., N. Luo, and G.H. Miley, *Cathode electrocatalyst selection and deposition for a direct borohydride/hydrogen peroxide fuel cell*. Journal of Power Sources, 2007. **173**(1): p. 77-85.
86. Zhang, L., et al., *Rapid and selective separation of iridium ions from aqueous solutions using nano-Al₂O₃*. Hydrometallurgy, 2012. **127**: p. 8-15.
87. Alexeyeva, N., et al., *Kinetics of oxygen reduction on gold nanoparticle/multi-walled carbon nanotube hybrid electrodes in acid media*. Journal of Electroanalytical Chemistry, 2010. **642**(1): p. 6-12.
88. Podlovchenko, B.I. and Y.M. Maksimov, *Open-circuit Potentials Established on Platinum and Gold Electrodes in PtCl₂-4 Solutions After the Displacement of Copper Adatoms*. Mendeleev Communications, 2013. **23**(3): p. 157-159.
89. Chen, W. and S.W. Chen, *Iridium-platinum alloy nanoparticles: Composition-dependent electrocatalytic activity for formic acid oxidation*. Journal of Materials Chemistry, 2011. **21**(25): p. 9169-9178.
90. Kjeang, E., et al., *Hydrogen peroxide as an oxidant for microfluidic fuel cells*. Journal of the Electrochemical Society, 2007. **154**(12): p. B1220-B1226.
91. *CRC handbook of chemistry and physics*. 1999, Chapman and Hall/CRCnetBASE: Boca Raton, FL.
92. reviewer, U., *Comments on "Modeling the Performance of an Ideal NaBH₄ – H₂O₂ Direct Borohydride Fuel Cell"*. 2013.
93. Lemmon, E.W., M.O. McLinden, and D.G. Friend, *Thermophysical Properties of Fluid Systems*, in *NIST Chemistry WebBook*, P.J. Linstrom and W.G. Mallard, Editors, National Institute of Standards and Technology: Gaithersburg MD, 20899.

94. Santos, D.M.F. and C.A.C. Sequeira, *Polymeric Membranes for Direct Borohydride Fuel Cells: a Comparative Study*. Alkaline Electrochemical Power Sources, 2010. **25**(13): p. 111-122.
95. Stroman, R.O. and G.S. Jackson, *Modeling the performance of an ideal NaBH₄-H₂O₂ direct borohydride fuel cell*. Journal of Power Sources, 2014. **247**: p. 756-769.