Modeling Membrane Degradation in Proton-Exchange-Membrane Fuel Cells

By

Victoria M. Ehlinger

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Chemical Engineering

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Dr. Adam Z. Weber, Co-Chair
Professor Jeffrey Reimer, Co-Chair
Professor Bryan McCloskey
Professor Van Carey

Fall 2020

Abstract

During operation, proton-exchange-membrane fuel cells (PEMFCs) are subjected to mechanical and chemical stressors that contribute to membrane degradation, performance loss, and eventual failure. Together, synergistic effects between mechanical and chemical degradation mechanisms lead to accelerated degradation. A physics-based model is developed to understand the synergistic effects of chemical and mechanical degradation and the coupled nature of performance and durability in PEMFCs. The model includes pinhole existence and growth in the membrane, which increases crossover of reactant gases as well as subsequent formation of chemical-degradation agents that impact both transport and mechanical properties of the membrane.

The underlying performance model accounts for the multi-component gas diffusion, reaction kinetics, and transport across the membrane. The membrane mechanical model assumes that a circular pinhole is present in the membrane and calculates the swelling strains and the elastic or plastic stresses on the pinhole. Additionally, an empirical model for the chemical degradation of the membrane via hydrogen peroxide and subsequent hydrogen fluoride generation is used to modify the mechanical properties as a result of chemical degradation. The fuel-cell model is fully coupled with a mechanical model to determine the stresses on the membrane and subsequent growth of pinholes during transient operation. Simulation results show pinhole growth under humidity cycling conditions and an increase in gas-crossover fluxes and decrease in performance. Sensitivity studies show how the membrane mechanical properties impact both the performance and degradation behavior of the membrane.

Multiphase effects are incorporated into the model to account for the effects of flooding on membrane degradation. Liquid condensation in the fuel cell can cause defects such as pinholes to close. Modeling results analyze the conditions under which water condensation will occur in pinholes, which is determined by calculating the critical radius. As the surface of Nafion can change from hydrophobic to hydrophilic, a sensitivity analysis on the critical angle is carried out. In addition, liquid water also reduces the amount of catalyst surface area available and therefore slows down the formation of hydrogen peroxide that drives chemical degradation. The decrease in chemical degradation at high RH values is demonstrated.

Cerium ions are added to the membrane to extend its lifetime by scavenging radicals produced by crossover of reactant gases during PEMFC operation. The cerium ions also lead to a decrease in performance due to changes in the PEM transport properties and possible site blockage in the catalyst layers. The PEMFC performance and durability model is extended to include micro-kinetic framework that accounts for gas-crossover-induced degradation and concentrated-solution theory describes transport in the PEM. The transport model takes into account the coupled nature of the electrochemical driving forces that cause transport of cerium ions, protons, and water. The cell model predicts the migration of cerium out of the membrane and into the catalyst layers and its impact on performance. A comparison of dilute-solution-theory and concentrated-solution-theory approaches illustrates the interactions between water and cerium transport in the cell. Transient simulations show that low concentrations of cerium in the membrane (less than 1% of membrane sulfonic acid sites occupied by cerium) are required to optimize these design tradeoffs.

Combining the mechanical and chemical degradation models, the mitigation effects of cerium on the coupled degradation methods can be shown. The model results show how the presence of a pinhole in the membrane shifts the distribution of cerium in the cell from the cathode into the anode and membrane. As the presence of cerium slows down the chemical degradation rate of the membrane, the rate of change of the mechanical properties of the membrane decreases. The model also shows how cerium modifies the mechanical and chemical degradation rates of the membrane under humidity- and voltage-cycling conditions.

Finally, three approaches for modeling the electrochemical impedance response of a PEMFC are compared using two case studies: a porous electrode with linear kinetics and a fuel cell cathode with Tafel kinetics. These approaches may be applied to the development of a physics-based electrochemical impedance model for a full fuel cell model. The first approach uses a transient-model approach, which is much slower and more prone to errors. However, this approach requires no additional modification of the time domain equations describing the system. The second and third approaches transform the transient model into frequency space and linearizing around the steady-state conditions. This approach is quick and accurate, but is impractical for highly coupled, nonlinear systems of equations. This approach applied to the fuel cell cathode with Tafel kinetics allows for analysis of system properties that change over time as a result of membrane degradation.

This thesis is dedicated to my husband Jeremy.

His love and support helped make this work possible.

Table of Contents

List of Figures

List of Tables

Acknowledgements

This work was made possible with the guidance and support of advisors, colleagues, family and friends. I would like to thank my fellow chemical engineering graduate students for help getting through many hours of coursework, studying for midterms and finals, and preparing for my preliminary and qualifying exams. I would like to thank my advisors Dr. Adam Weber and Dr. Jeffrey Reimer for their mentorship throughout my PhD. I would like to thank all of my colleagues in the Energy Conversion Group at Lawrence Berkeley National Laboratory, for many helpful discussions and for being an excellent group of colleagues to work with, both in terms of the rigor of academic research and creating a welcoming and supportive lab environment. Thanks especially to Ahmet Kusoglu for his guidance with the mechanical properties and modeling of membranes, Lalit Pant for his guidance on modeling and multiphase phenomena in PEM fuel cells, Andrew Crothers for his guidance on thermodynamics and transport, Hans Johansen for his guidance on mathematical solvers and algorithms, and Arthur Dizon for his guidance on electrochemical impedance modeling. Lastly, I would like the Department of Energy Fuel Cell Consortium for Performance and Durability for funding for this work.

Chapter 1 – Introduction

Proton-exchange-membrane fuel cells (PEMFCs) are electrochemical devices that convert hydrogen fuel to electricity, with water as the only byproduct. PEMFCs have applications in stationary power, portable power, and transportation. With increasing concerns regarding climate change and pollution, PEMFCs are a promising clean-energy technology. However, advances in PEMFC performance and durability and cost reduction are necessary in order to compete with traditional internal-combustion engines.

According to the U.S. Department of Energy, for light-duty vehicle applications a PEMFC must be able to operate for 5,000 hours with less than 10% performance loss.[1] Transportation applications impose additional durability challenges over stationary power applications, where the fuel cell can be run continuously at steady-state conditions. Under operation in a vehicle, the fuel cell must also be able to withstand start/stop cycles, load cycles (changes in power demand on the engine due to acceleration/deceleration), and changes in environmental conditions such as relative humidity (RH) and temperature (including sub-freezing temperatures). These transient operating conditions are some of the primary driving forces for membrane degradation in PEFMCs, motivating the need to better understand the underlying mechanisms behind membrane degradation and how to mitigate them.

1.1 PEMFC Operation

A schematic of a polymer electrolyte fuel cell is shown in Figure 1-1. Hydrogen fuel and air enter the fuel cell through the gas channels, with hydrogen fuel fed on the anode side and oxygen/air fed on the cathode side. As gases flow through the gas channels, they diffuse across the cell through the gas diffusion layers, which serve to distribute the feed gases across the cross-sectional area of the fuel cell. The gas diffusion layers are typically made of a carbon paper or carbon fiber. Once the gases exit the gas diffusion layers, they enter the catalyst layers where the electrochemical reactions are carried out on a platinum catalyst. In the anode the hydrogen oxidation reaction (HOR) occurs to generate protons and electrons from hydrogen gas.

$$2\mathrm{H_2} \rightarrow 4\mathrm{H^+} + 4\mathrm{e^-} \qquad (1.1)$$

The protons are then selectively transported across the PEM and the electrons flow through an external current to perform work. In the cathode, the oxygen reduction reaction (ORR) occurs when oxygen in the air reacts with protons and electrons to form water,

$$\mathrm{O_2} + 4\mathrm{H^+} + 4\mathrm{e^-} \rightarrow 2\mathrm{H_2O} \qquad (1.2)$$

Perfluorosulfonic acid (PFSA) membranes are typically used in PEMFCs. These ionomers consist of a fluorocarbon polymer-chain backbone and side chains terminated by a sulfonic-acid group. The negatively charged acid sites allow for protons to be selectively transported through the membrane. The hydration of the membrane plays an important role in the performance of the fuel cell, as higher hydration improves ionic conductivity through the membrane. The water content (λ) of the membrane is defined as the number of water molecules per sulfonic-acid group.

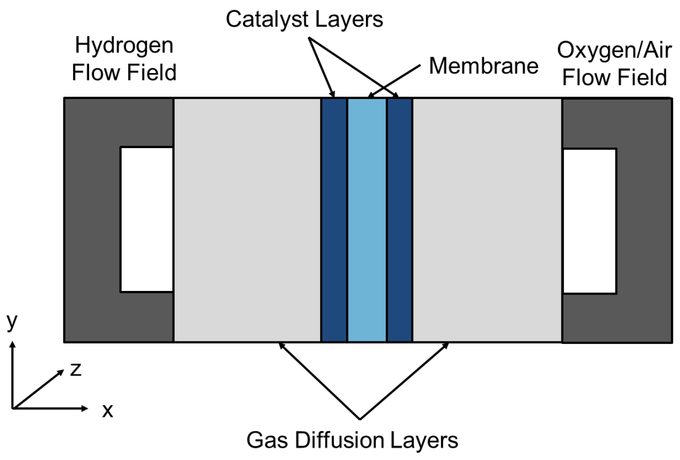


Figure 1-1: Schematic showing the various layers of the fuel-cell sandwich.

The catalyst layers consist of platinum nanoparticles embedded on a carbon support. These catalyst particles are surrounded by ionomer, which is typically the same ionomer as used in the membrane. The catalyst-layer particles form agglomerates, such that reactive gases must diffuse through the ionomer film and the agglomerate in order to react at the catalyst surface. Illustrations of the chemical structure of Nafion, a hydrated PFSA membrane, and catalyst-layer particles are shown in Figure 1-2.

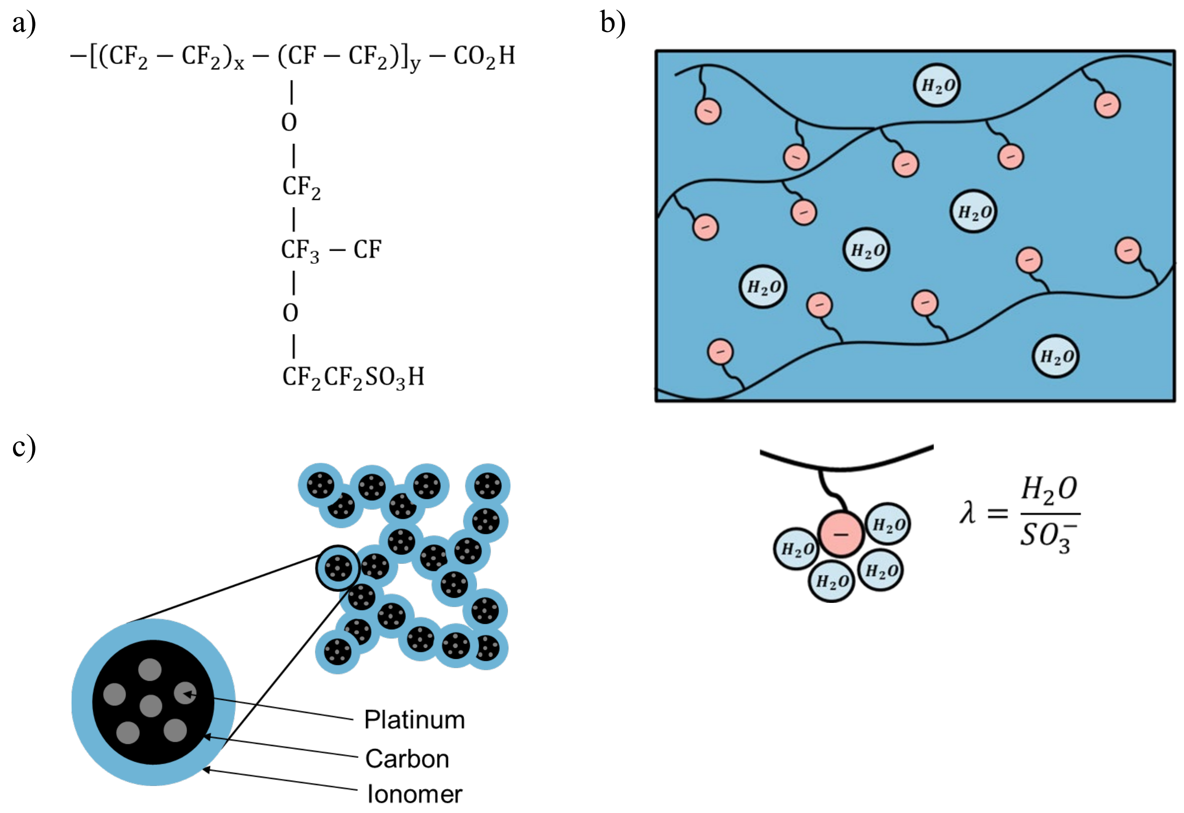

a)
$-[(CF_2-CF_2)_x-(CF-CF_2)]_y-CO_2H$
O
CF_2
CF_3-CF
O
$CF_2CF_2SO_3H$
b)
H_2O
$\lambda = \frac{H_2O}{SO_3^-}$
c)
Platinum
Carbon
Ionomer

Figure 1-2: Illustration of fuel cell components, a) chemical formula of Nafion, b) water content in the membrane, c) catalyst layer agglomerate structure.

1.2 Fuel-Cell Degradation Mechanisms

With increasing interest in PEMFCs for medium- and heavy-duty applications, such as buses and long-haul trucks, research efforts have shifted towards improving lifetime and durability to enable commercialization. During PEMFC operation, mechanical and chemical stressors occur in the membrane, leading to loss of performance, or even catastrophic failure of the PEM. Mechanical degradation is driven by swelling strain inside the membrane while under hydration cycles, which causes the formation and growth of defects (e.g. pinholes). Chemical degradation is driven by the formation of highly reactive hydroxyl radicals, which react with the ionomer and causes loss of conductivity and increased gas crossover. Synergistic interactions between mechanical and chemical degradation mechanisms can cause degradation to accelerate over time.

Fuel-cell membranes undergo mechanical and chemical stressors during operation, which leads to degradation, performance loss, and eventual failure. Failure could occur in the form of formation and growth of defects, delamination, membrane thinning, etc.[2-6] Figure 1-3 illustrates various degradation mechanisms that may occur in polymer-electrolyte membranes during fuel cell operation.

Mechanical degradation typically occurs due to stresses acting on manufacturing defects in the membrane or sites initiated by chemical attack during operation.[2, 4, 5, 7-11] As the RH changes during fuel-cell operation, swelling and deswelling of a membrane constrained in the cell results in varying stress states and eventually permanent (plastic) deformation. This deformation leads to formation of defects such as pinholes and cracks, and causes growth of defects over operational time.[12, 13]

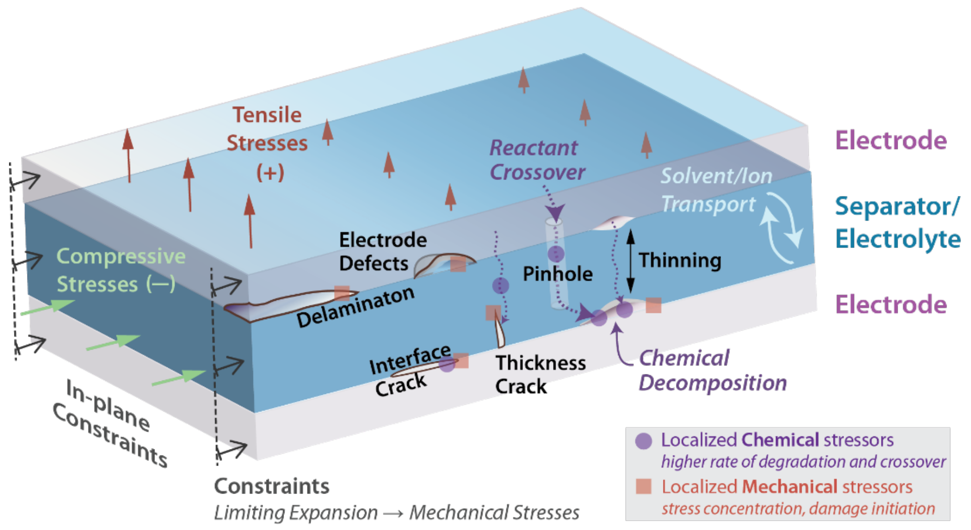


Figure 1-3: Degradation modes in polymer-electrolyte membranes. Reprinted with permission from Kusoglu and Weber.[3]

Chemical degradation results from the attack of hydroxyl radicals generated by the decomposition of H_2O_2, which is formed by the two-electron oxygen reduction reaction upon the crossover of the reactant gases (H_2 and O_2) through the membrane.[14, 15]

$$O_2 + 2e^- + 2H^+ \rightarrow H_2O_2 \quad (1.3)$$

Hydroxyl radicals are also generated via Fenton's reaction of H_2O_2 with iron ions, which is believed to be present in the PEM due to migration from the metallic bipolar plates.[16, 17]

$$H_2O_2 + Fe^{2+} + H^+ \rightarrow Fe^{3+} + OH^{\bullet} + H_2O \quad (1.4)$$

These radicals attack the chemical bonds in the ionomer's fluorocarbon backbone and side-chains and causes loss of membrane structure and integrity, thereby impacting its properties.[18-20] In addition, reactive gases can crossover through the membrane and react at the opposite electrode, leading to a mixed potential on the electrodes and an overall loss of power output. Furthermore, the reaction of these crossover gases are highly exothermic and can create hotspots that lead to further thermal decomposition of the membrane material.[13]

As membrane defects grow as a result of swelling stresses, the gas crossover through the membrane may further increase, leading to a self-propagating cycle of membrane degradation, as illustrated in Figure 1-4, until the PEMFC fails due to chemical shorting. Increased gas crossover through membrane defects leads to increased radical formation, which leads to localized radical attack on the surrounding polymer and further defect growth.[4]

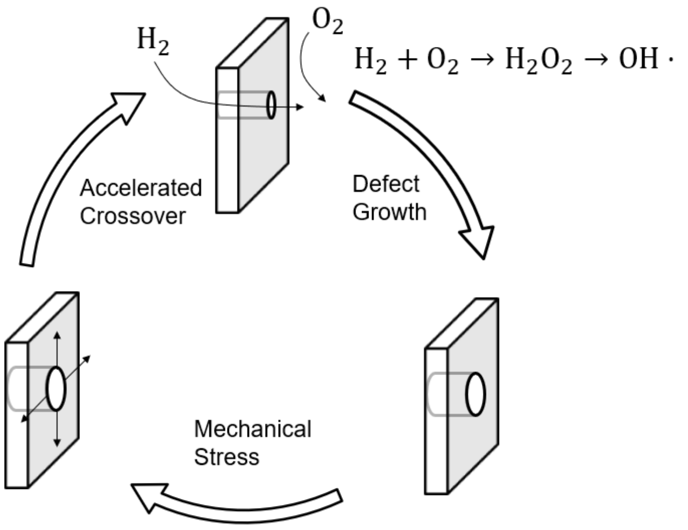


Figure 1-4: Membrane degradation mechanisms and feedback. Adapted with permission from Kusoglu and Weber [3].

Figure 1-5 summarizes the driving forces and synergistic interactions between various degradation modes in PFSA membranes.

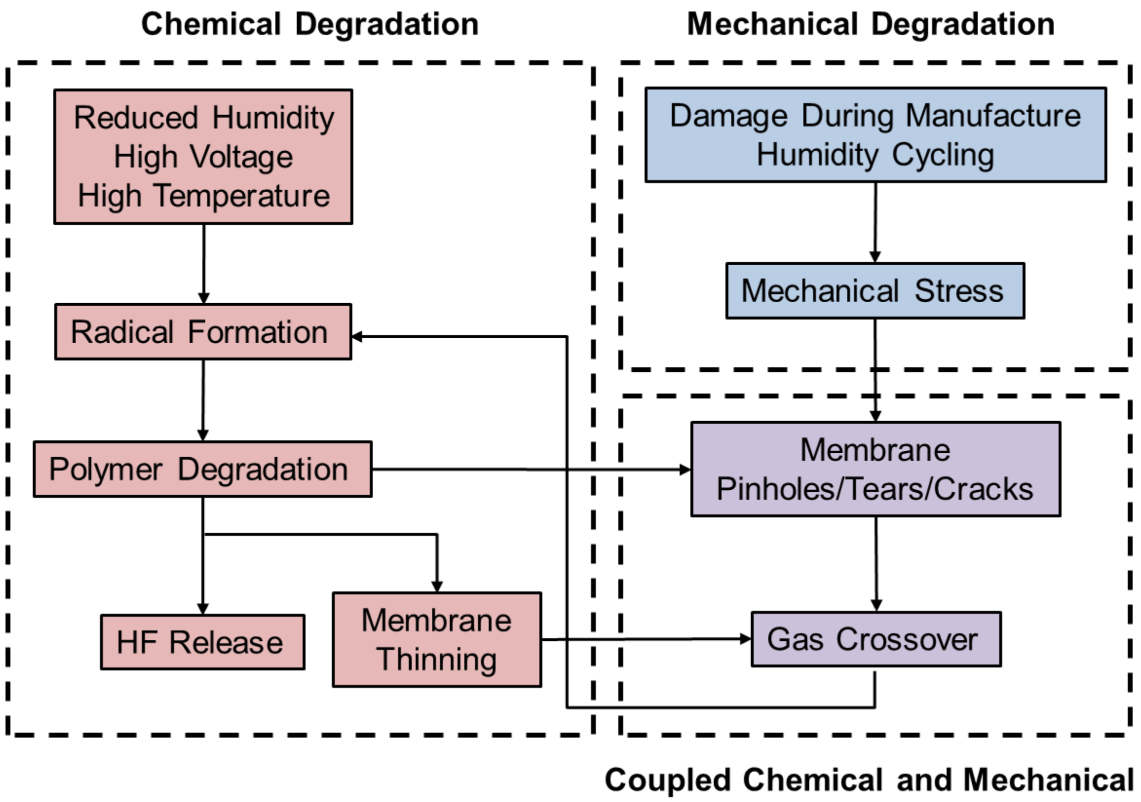


Figure 1-5: Key physical phenomena driving coupled chemical and mechanical membrane degradation in PEFCs during operation.

A number of experimental protocols have been developed to characterize the degradation of PEMs. Often these experimental protocols are referred to as accelerated stress tests (ASTs), as the purpose is to replicate the degradation phenomena during typical operating conditions in a lab setting on a time scale faster than the hundreds or thousands of hours required for a fuel cell to fail. ASTs are often used to accelerate a particular stressor, so they may be categorized as mechanical, chemical, or combined chemical/mechanical. Drive cycles, which have been adapted from similar tests for internal-combustion engines, subject the fuel cell to a series of varying power demands. Mechanical durability tests are carried out by subjecting the fuel cell to a series of RH cycles while monitoring gas crossover. Chemical durability tests are carried out using an open circuit voltage (OCV) hold while monitoring the fluoride release rate (FRR), gas crossover, and change in OCV. Hydrogen fluoride can be detected in the fuel-cell effluent, which is released during chemical degradation as fluorocarbon bonds in the ionomer are broken. OCV decay indicates a permanent loss in power output from the cell. A test developed by General Motors for chemical degradation runs the cell at OCV conditions, 50% RH, and high temperature (95°C), demonstrates rapid FRR acceleration and can be carried out in less than 200 hours.[2] Additionally, Fenton's test is a commonly used *ex-situ* test for chemical durability, which involves using a hydrogen peroxide solution in the presence of iron II ions. Combined chemical/mechanical durability tests involve a combination of both mechanical and chemical stressors, such as humidity cycling, current cycling, OCV operation, and high temperatures.[5] These tests sometimes alternate between chemical and mechanical degradation modes and sometimes apply both types of stressors simultaneously.

While membrane degradation is a primary cause for PEM fuel cell failure, degradation in other components of the cell contribute to loss of performance. Additionally, cycling conditions that drive membrane degradation can also drive other modes of degradation in different parts of the cell. An overview of degradation mechanisms in PEMFCs is given by Borup *et al*.[21]

1.3 Degradation Mitigation

To improve mechanical durability, a reinforcing layer such as expanded polytetrafluoroethylene (ePTFE) can be incorporated. To reduce chemical degradation in PEMFCs, radical scavengers such as Ce^{3+} and Mn^{2+} and their metal oxides are embedded into the membrane. The purpose of these radical scavengers is to react with the hydroxide radicals before attacking the membrane due to more favorable thermodynamics and faster reaction kinetics. Experiments have shown that cerium acts as a highly effective mitigant of membrane degradation in PEMFCs.[22-24] However, increasing concentrations of cerium in the PEM decrease its proton conductivity and inhibit the oxygen-reduction-reaction kinetics due to a lower availability of protons, which can result in proton limiting currents.[6, 25-28]

1.4 Performance vs. Durability

Performance and durability are often seen as competing targets, since many approaches for improving performance lead to decreased durability and vice versa. Higher temperatures and humidity result in increased conductivity, but cause the membrane to degrade faster.[2] Membranes with lower equivalent weight also improve conductivity, but undergo greater swelling strains as a result of water uptake.[2] Thinner membranes provide lower ohmic resistance, but have higher gas crossover, which causes increased voltage decay rates and earlier membrane failure. Both supportive polymer layers and chemical scavengers added to the membrane to mitigate degradation cause a decrease in conductivity. Optimization of PEMFC design and operation requires understanding of the underlying physical interactions that drive degradation. Multiphysics models are one approach to understanding the tradeoffs between competing design criteria for performance and durability.

1.5 Modeling and Fuel Cell Degradation

Recent modeling studies have been carried out to better understand the mechanisms behind PEMFC degradation and how to mitigate the underlying causes to improve durability. A variety of modeling approaches have been used to study membrane degradation on the continuum scale as well as the molecular scale. A modeling approach to degradation provides several advantages. First, experimental ASTs take a long time to carry out, on the order of hundreds of hours. A model allows for sensitivity studies of different material properties and operating conditions to be carried out quickly and easily. Some of the transport properties of the system, particularly ion transport properties within in membrane, cannot be easily measured, but may be calculated using a theoretical model. Previous modeling efforts on membrane degradation in PEMFCs focus on one method of degradation, either mechanical or chemical. However, the synergistic interactions between these two degradation modes have been repeatedly demonstrated. The purpose of this work is to develop a continuum-level mathematical model that incorporates both chemical and mechanical degradation modes in order to understand how and why the synergistic interactions occur within the system and optimize PEMFC design and structure for both performance and durability. While some chemical degradation studies have been carried out using a molecular modeling approach[29-31] (i.e. molecular dynamics, Monte Carlo), this approach is not well suited to coupling the mechanical and chemical degradation phenomena. Continuum-level modeling allows for the full cell to be modelled and can include the membrane degradation kinetics as well as the mechanical model for stresses acting on the membrane.

PEMFC models can be categorized by their dimension, which are indicated by the axes in Figure 1-1. The simplest models are referred to as 0-D, which describe the fuel cell performance with a single equation, typically fit to experimental data. An example of a 0-D PEMFC model is a polarization equation, which calculates potential as a function of current density. 1-D PEMFC models are in the x-direction in Figure 1-1, which is referred to as the fuel-cell sandwich. 2-D models can be in the x-direction and either the y-direction (across-the-gas-channel) or the z-direction (along-the-channel). A full 3-D model would consist of the fuel-cell sandwich, across-the-channel, and along-the-channel directions, and are generally very computationally expensive.

1.5.1 Mechanical Degradation Models

Solasi *et al.* modeled the expansion/contraction mechanical response of an ionomer membrane under thermal and hydration cycling using a 2-D finite element model.[32] Their results showed that hydration had a larger effect on the membrane stress response than temperature, and that the maximum stress and strain values were present at the edges of the membrane. Kusoglu and Weber developed a 0-D model to describe the expansion of an idealized pinhole in a fuel cell membrane.[12] The model is able to determine based on the amplitude of a hydration cycle, whether the swelling stress will cause plastic or elastic deformation. In cases where plastic deformation occurs, a permanent increase in the pinhole size occurs, describing how the pinhole grows during mechanical AST conditions. Hasan *et al.* investigated the effect of RH cycling of a fuel cell on the elastic-plastic response of the membrane using a transient, 2-D finite element model of the unit cell.[33] The results show that the areas of maximum tensile stress in the membrane vary based on temperature, and that in all cases the maximum compressive stresses occur under the land near the cathode.

1.5.2 Chemical Degradation Models

Gubler and coworkers developed a 0-D model for radical formation as a result of Fenton's reaction and radical attack on fuel cell membranes, and also incorporated the effects of adding cerium and manganese ions as chemical scavengers.[34, 35] Their approach enabled them to replicate results from *ex-situ* Fenton tests as well as make predictions for chemical degradation for *in-situ* conditions. Wong and Kjeang incorporated the kinetics of Gubler's 0-D model into a 1-D fuel cell model with chemical degradation.[36, 37] They later investigated the effects of ceria additives and analyzed tradeoffs between performance and durability.[38, 39] Their results show that the concentration of H_2O_2 is highest in the anode catalyst layer and that operating at a potential below 0.7 V decreases the FRR by an order of magnitude. They also showed that the presence of cerium ions in the membrane leads to a decrease in performance due to increased ohmic and kinetic potential losses. Singh *et al.* developed a transient, 2-D model that incorporates chemical-degradation phenomena along with fuel-cell performance.[40] The model predicts how the membrane degrades into various fragments and releases hydrogen fluoride as a result. The results were validated by comparison of polarization curves and FRR rates during an OCV hold with experimental data. Futter *et al*. conducted a modeling study to analyze the effect of operating conditions including pressure, RH, and cell voltage on the chemical degradation rates of fuel cell membranes driven by radical formation as a result of the presence of iron contaminants.[41] Their results show that at high pressure, high RH, and high potential the chemical degradation of the membrane accelerates; therefore, chemical degradation can be partially mitigated by operating the cell under different conditions.

1.6 Thesis Objectives and Outline

The purpose of this thesis is to develop a mathematical model of combined mechanical and chemical degradation in PEMFCs in order to probe the synergistic interactions between degradation modes. Additionally, a model for cerium transport and chemical degradation is used to analyze the impact of chemical scavengers. Chapter 2 introduces the underlying performance model for the PEMFC, which is transient and 1D across the fuel-cell sandwich. This chapter outlines transport and kinetics equations as well as the physical parameters used in the model. Chapter 3 introduces the mechanical model used to describe a circular pinhole inside the membrane and its growth as a result of hydration and swelling cycles. The mechanical model is coupled with the performance model and the impact of mechanical degradation on fuel-cell performance is detailed. Chapter 4 adds multiphase phenomena to the fuel cell performance model and demonstrates the effects of flooding on membrane degradation. Chapter 5 incorporates a microkinetic model for chemical degradation with the fuel-cell performance model. A concentrated-solution-theory approach is used to account for the transport of cerium ions within the ionomer. The model allows for an analysis of a tradeoff performance and durability metrics while chemical scavengers are used in the cell. Chapter 6 describes the methods for deriving a physics-based impedance model that can be applied to the PEMFC model. Lastly, a summary of findings and directions for future work are outlined in Chapter 7.

Chapter 2 – Fuel Cell Model

The fuel-cell performance model is based upon the work of Fuller and Newman[42, 43] and subsequent work of Weber and Newman.[44-48] The model is transient and 1-D across the fuel cell sandwich (x-direction in Figure 1-1). Stefan-Maxwell equations are used for transport of gaseous species through the porous media. Butler-Volmer kinetics are used for the hydrogen oxidation reaction (HOR) and oxygen reduction reaction (ORR). The current density through the solid phase (i_1) is governed by Ohm's law. The current density through the membrane phase (i_2) as well as water transport through the membrane are derived from concentrated-solution theory, assuming the membrane is an isothermal, isotropic mixture of water, protons, and sulfonic acid sites.

2.1 Porous Media

In the solid phase, Ohm's law holds,

$$\mathrm{i}_1 = -\sigma^{\mathrm{eff}} \nabla \Phi_1, \tag{2.1}$$

where i_1 and Φ_1 are the current and potential in the solid phase, respectively, and σ^{eff} is the effective bulk electronic conductivity. In addition, current conservation holds throughout the entire PEMFC domain and double-layer charging is neglected.

$$\nabla \cdot \mathrm{i}_1 + \nabla \cdot \mathrm{i}_2 = 0 \tag{2.2}$$

In the gas-diffusion layers and catalyst layers, Stefan-Maxwell equations are used to describe the multi-component diffusion,

$$\nabla p_i = -\frac{p_i}{RT}\left(\bar{V}_i - \frac{M_i}{\rho}\right)\nabla p + \sum_{\mathrm{j}=1}^{\mathrm{n}} \frac{y_i \mathbf{N}_{\mathrm{j}} - y_j \mathbf{N}_{\mathrm{i}}}{c_T \mathcal{D}_{ij}^{\mathrm{eff}}} - \frac{\mathbf{N}_{\mathrm{i}}}{c_T \mathcal{D}_{K_i}^{\mathrm{eff}}}, \tag{2.3}$$

where p is the gas pressure, T is the temperature, R is the ideal gas constant, ρ is the gas density, p_i, $\bar{V}_i$, M_i, y_i, $\mathbf{N}_{\mathrm{i}}$ are the partial pressure, partial molar volume, molecular weight, mole fraction, and flux of species i (N_2, H_2, H_2O, or O_2), respectively, c_T is the total gas concentration, $\mathcal{D}_{ij}^{\mathrm{eff}}$ is the effective diffusion coefficient between species i and species j, and $\mathcal{D}_{K_i}^{\mathrm{eff}}$ is the effective Knudsen diffusion coefficient. The Knudsen diffusion correction becomes important when the mean free path of the diffusing molecules is similar in magnitude to the characteristic pore size for gas transport, which is true for catalyst layers. Diffusion coefficients and physical properties of water are given in Table 2-1. The electrode specific interfacial area is a fitting parameter as this value is typically not known. A value of 1×10^{-5} cm^{-1} is used for the simulations in Chapters 3-4 and a value of 8×10^{-5} cm^{-1} is used in Chapter 5 due to fitting to different data sets. Effective diffusion coefficients and other transport properties are calculated using the values in Table 2-1 and applying the Bruggeman correction. The Bruggeman correction states that for any transport property Y, the effective value in a porous medium is given by,

$$Y^{\mathrm{eff}} = Y\frac{\varepsilon}{\tau} \tag{2.4}$$

where ε is the void fraction and τ is the tortuosity. Void fractions and membrane volume fractions for each layer are listed in Table 2-2.

Pressure drop across the fuel-cell sandwich is calculated using Darcy's Law,

$$\mathbf{v} = -\frac{k}{\mu}\nabla p \tag{2.5}$$

where k is the gas permeability and μ is the gas viscosity.

Table 2-1: Physical Properties

Property		Units	Equation	Ref
Water density	ρ_w	g/cm^3	$1.1603 - 5.371 \times 10^{-4}T$	46
Water viscosity	μ_w	bar·s	$1 \times 10^{-11}(2695.3 - 6.6T)$	46
Water vapor pressure	p_w^{vap}	bar	$\exp\left(11.6832 - \frac{3816.44}{T - 46.13}\right)$	46
Water surface tension	γ	N/m	$0.12398 - 1.7393 \times 10^{-4}T$	47
Hydrogen/water diffusion coefficient	$p\mathcal{D}_{H_2,w}$	bar·cm^2/s	$2.470\left(\frac{T}{146.55}\right)^{2.334}$	46
Oxygen/water diffusion coefficient	$p\mathcal{D}_{O_2,w}$	bar·cm^2/s	$0.3022\left(\frac{T}{323.83}\right)^{2.334}$	47
Nitrogen/oxygen diffusion coefficient	$p\mathcal{D}_{N_2,O_2}$	bar·cm^2/s	$0.0544\left(\frac{T}{143.01}\right)^{1.823}$	47
Electrode specific interfacial area	a	cm^{-1}	$10^5 - 10^6$	fit
Membrane/water vapor rate constant	$k_{M,V}$	$\frac{\text{mol}^2}{\text{s} \cdot \text{J} \cdot \text{cm}^3}$	10^5	fit
Membrane/liquid water rate constant	$k_{M,L}$	$\frac{\text{mol}}{\text{s} \cdot \text{bar} \cdot \text{cm}^3}$	1000	fit
Water evaporation constant	k_{evap}	$\frac{\text{mol}^2}{\text{s} \cdot \text{J} \cdot \text{cm}^2}$	100	fit

Table 2-2: Fuel Cell Transport Properties

Property		Units	Gas Diffusion Layers	Catalyst Layers	Membrane	Ref.
Thickness	L	cm	0.025	0.002	0.0025	47
Volume fraction for gas transport	ε_0		0.6	0.3	0	13, 47
Volume fraction of membrane	ε_M		0	0.4	1	13, 47
Absolute permeability	k	cm^2	6e-8	8e-12	1.8e-14	49
Bulk-phase conductivity	σ	S/cm	7.14	8.4	0	47, 50
Effective thermal conductivity	k	W/(cm·K)	0.015	0.003	0.0025	48

An energy balance is used to determine the temperature profile across the fuel cell,

$$\rho\hat{C}_p\left(\frac{\partial T}{\partial t}+\mathbf{v}\cdot\nabla \mathrm{T}\right)-\nabla\cdot(k\nabla T)=\frac{\mathbf{i}\cdot\mathbf{i}}{\kappa}+\sum_{\mathrm{h}}\mathrm{i}_{\mathrm{h}}(\eta_h+\Pi_h) \tag{2.6}$$

The first term on the left represents the transport and accumulation of enthalpy, where $\hat{C}_p$, ρ, and $\mathbf{v}$ are the average heat capacity, density, and mass-averaged velocity, respectively. The second term on the left is heat transfer by conduction, where k is the thermal conductivity (see Table 2-2). The right side of the equation represents the heat generation and consumption terms. The first term on the right side is Joule or ohmic heating. The second term is generation due to electrochemical reactions, where η_h and Π_h are the overpotential and Peltier coefficient of reaction h, respectively.

2.2 Conservation Equations

A conservation-of-mass equation is necessary to account for changes in species fluxes due to transient phenomena and reaction rates:

$$\frac{dc_i}{dt}+\nabla\cdot\mathbf{N}_\mathrm{i}=R_i \tag{2.7}$$

where c_i and R_i are the concentration and reaction rate of species i, respectively. For an electrochemical reaction, the rate of reaction can be expressed as a function of the current generated,

$$R_i=-\frac{a}{n_hF}\mathbf{i}_\mathrm{h}E_h \tag{2.8}$$

where $\mathbf{i}_\mathrm{h}$, n_h, and E_h are the current density, number of electrons, and effectiveness factor of reaction h, respectively, and a is the specific electrode interfacial area.

2.3 Kinetics

For the hydrogen oxidation reaction (HOR) at the anode, Bulter-Volmer kinetics are used. For the oxygen reduction reaction (ORR) at the cathode, Tafel kinetics are used because the cathodic term dominates due to the sluggishness of the reaction.

$$\begin{aligned}\mathbf{i}_\mathrm{HOR}=i_{0_\mathrm{HOR}}\Bigg[\frac{p_{H_2}}{p_{H_2}^\mathrm{ref}}\exp\left(\frac{\alpha_aF}{RT}\left(\Phi_1-\Phi_2-U_0^\mathrm{HOR}\right)\right)\\-\exp\left(-\frac{\alpha_cF}{RT}\left(\Phi_1-\Phi_2-U_0^\mathrm{HOR}\right)\right)\Bigg]\end{aligned} \tag{2.9}$$

$$\mathbf{i}_\mathrm{ORR}=-i_{0_\mathrm{ORR}}\frac{p_{O_2}}{p_{O_2}^\mathrm{ref}}\exp\left(-\frac{\alpha_cF}{RT}\left(\Phi_1-\Phi_2-U_0^\mathrm{ORR}\right)\right) \tag{2.10}$$

where $\mathrm{i}_{0,\mathrm{HOR}}$ and $\mathrm{i}_{0,\mathrm{ORR}}$ are the respective exchange current densities, α_a and α_c are the anode and cathode coefficients, U_0^HOR and U_0^ORR are the respective standard potentials versus the standard hydrogen electrode (SHE), R is the ideal gas constant, and T is the absolute temperature. The two-electron oxygen reduction reaction, which produces hydrogen peroxide, can be written in a similar manner as Equation (2.10). Note that the reactions can occur on either electrode since crossover is considered. The kinetic parameters are shown in Table 2-3.

An agglomerate model is used to take into account the effects of gases diffusing into the catalyst particles in order to react. To achieve an easily calculated solution, the catalyst particles are assumed to be spherical, although in PEMFCs catalyst layer agglomerates vary in shape and size. With first-order kinetics and assuming a spherical catalyst particle, the effectiveness factor for the agglomerate is,

$$E_h = \frac{1}{3\phi^2}(3\phi \coth(3\phi) - 1) \tag{2.11}$$

where ϕ is the Thiele modulus, which is defined as the ratio of the rate of reaction over rate of mass transport into the catalyst particle,

$$\phi = \sqrt{\phi_{mt} k_h{}'} \tag{2.12}$$

where ϕ_{mt} is the mass transport term and k_h' is the kinetic term for reaction h. The kinetic terms are given by

$$k'_{\text{HOR}} = \frac{a i_{0,\text{HOR}}}{2F p_{H_2}^{ref}} \exp\left(\frac{\alpha_a F}{RT}\left(\Phi_1 - \Phi_2 - U_0^{\text{HOR}}\right)\right) \tag{2.13}$$

$$k'_{\text{ORR}} = \frac{a i_{0,\text{ORR}}}{4F p_{O_2}^{\text{ref}}} \exp\left(-\frac{\alpha_c F}{RT}\left(\Phi_1 - \Phi_2 - U_0^{\text{ORR}}\right)\right) \tag{2.14}$$

The Thiele mass transport coefficient for the two-electron oxygen reduction reaction is 6000 bar·cm·s/mol in Chapters 3-4 and E = 1 in Chapter 5.

Table 2-3: Kinetic Properties

Property		**Units**	**HOR**[47]	**$4e^-$ORR**[47]	**$2e^-$ORR**
Activation energy	E_A	J/mol	9500	73269	—
Exchange current density	i_0	A/cm^2	$10^{-4}\left(\frac{E_A}{R}\left(\frac{1}{T}-\frac{1}{T_{ref}}\right)\right)$	$1.1\times10^{-8}\left(\frac{E_A}{R}\left(\frac{1}{T}-\frac{1}{T_{ref}}\right)\right)$	7×10^{-7}
Equilibrium potential	U_0	V	0	$4.1868((70650 + 8T\log T - 92.4\,T)/2\text{F})$	0.695
Anodic transfer coefficient	α_a		1	—	—
Cathodic transfer coefficient	α_c		1	1	1
Thiele mass transport coefficient	ϕ_{mt}	$\frac{\text{bar}\cdot\text{cm}\cdot\text{s}}{\text{mol}}$	8000	6000	6000 or $E = 1$

2.4 Membrane Transport

The underlying membrane model is based on concentrated-solution theory for protons and water mass transport in the PEM. In terms of measurable quantities, the governing transport equations are

$$\mathbf{i}_2 = -\kappa\nabla\Phi_2 - \frac{\kappa\xi}{F}\nabla\mu_w \tag{2.15}$$

$$\mathbf{N}_\mathrm{w} = -\frac{\kappa\xi}{F}\nabla\Phi_2 - \left(\alpha + \frac{\kappa\xi^2}{F^2}\right)\nabla\mu_w, \tag{2.16}$$

where $\mathbf{N}_\mathrm{w}$ is the flux of water, $\mathbf{i}_2$ is the current in the electrolyte phase, Φ_2 is the potential in the electrolyte phase, F is Faraday's constant, μ_w is the (electro)chemical potential of water, κ is the ionic conductivity, ξ is the electroosmotic coefficient, and α is the membrane transport coefficient. The membrane material properties used are from the works of Weber and Newman.[45, 46, 48]

During PEMFC operation, some of the reactant gases can diffuse through the membrane and react at the other electrode. The rate of gas crossover through the membrane is given by

$$\mathbf{N}_\mathrm{i} = -\psi_i\nabla p_i \tag{2.17}$$

where ψ_i and p_i are the permeation coefficient and partial pressure of species i, respectively. This is essentially a combination of Henry's and Fick's laws for the gases in the membrane and the coefficients are taken from experimental data.[45]

Several of the key membrane-transport properties depend on the water content of the membrane. The water content, λ_V, is defined as the moles of water per mole of sulfonic acid sites in the membrane. To calculate the water content, Kusoglu and Weber reported a polynomial fit to over multiple sets of literature data,[51]

$$\lambda_\mathrm{V} = 0.05 + 20.45\mathrm{a} - 42.8\mathrm{a}^2 + 36\mathrm{a}^3, \tag{2.18}$$

where a is the water activity. Expressions for the membrane properties in terms of water content and temperature are summarized in Table 2-4. These expressions are valid for a vapor-equilibrated membrane (no liquid water is present in the cell).

2.5 Model Solution

The fuel cell performance model is run in MATLAB (see Appendix B for codes used). This model serves as the foundation upon which the additional physics of chemical and mechanical degradation are built upon. In order to run the model, fuel-cell operating conditions and physical properties must be specified. For example, operating conditions include temperature, pressure, air and feed flow rates or stoichiometry, applied voltage or potential. Physical properties include solid-phase conductivity, equivalent weight of the membrane, and fuel-cell layer thicknesses. The governing equations are discretized and fluxes are calculated using a finite volume approach. A full list of governing equations and boundary conditions are listed in Appendix A. The system of governing equations is then solved using the BAND(J) algorithm.[52, 53] A Crank-Nicolson approach is used for time discretization of transient equations. Chapters 3-5 contain a more detailed description of the solution approach used for the each of the degradation models described.

Table 2-4: Vapor-Equilibrated Membrane Property Calculations [45]

Parameter		Units	Equation
Membrane Water Vapor Volume Fraction	f_V		$f_V = \frac{\lambda_V \bar{V}_0}{\bar{V}_m + \lambda_V \bar{V}_0}$
Conductivity	κ_V	$\mathrm{S/cm}$	if $f_V \le 0.45$ $\kappa_V = \frac{1}{2}(f_V - 0.06)^{1.5} \exp\left(\frac{15000}{R}\left(\frac{1}{T_{ref}} - \frac{1}{T}\right)\right)$ if $f_V > 0.45$ $\kappa_L = \frac{1}{2}(0.39)^{1.5} \exp\left(\frac{15000}{R}\left(\frac{1}{T_{ref}} - \frac{1}{T}\right)\right)$
Electroosmotic coefficient	ξ_V		$\xi_V = \lambda_V$ if $\lambda_V < 1$ $\xi_V = 1$ if $\lambda_V \ge 1$
Water membrane diffusion coefficient	$\mathcal{D}_{\mu 0}$	$\frac{\mathrm{cm}^2}{\mathrm{s}}$	$\mathcal{D}_{\mu 0} = 1.8 \times 10^{-5} f_V \exp\left(\frac{20000}{R}\left(\frac{1}{T_{ref}} - \frac{1}{T}\right)\right)$
Membrane transport coefficient	α_V	$\frac{\mathrm{mol}^2}{\mathrm{J \cdot cm \cdot s}}$	$\alpha_V = \frac{C_{0V} D_{\mu 0}}{RT(1 - x_{0V})}$
Hydrogen gas permeation coefficient	$\psi_{H_2,V}$	$\frac{\mathrm{mol}}{\mathrm{bar \cdot cm \cdot s}}$	$\psi_{H_2,V} = (2.2 \times 10^{-11} f + 2.9 \times 10^{-12}) \exp\left(\frac{21000}{R}\left(\frac{1}{T_{\mathrm{ref}}} - \frac{1}{T}\right)\right)$
Oxygen gas permeation coefficient	$\psi_{O_2,V}$	$\frac{\mathrm{mol}}{\mathrm{bar \cdot cm \cdot s}}$	$\psi_{O_2,V} = (1.9 \times 10^{-11} f + 1.1 \times 10^{-12}) \exp\left(\frac{22000}{R}\left(\frac{1}{T_{\mathrm{ref}}} - \frac{1}{T}\right)\right)$

Chapter 3 – Synergistic Mechanical and Chemical Degradation

In order to analyze the effects of mechanical membrane degradation during operation, the fuel cell performance model is coupled with a membrane mechanical degradation model. The model assumes that a circular pinhole is present in the membrane, which allows for crossover gasses to diffuse through. A mechanical model, initially developed by Kusolgu and Weber,[12] is used to calculate the swelling strain due to water uptake in the membrane and determine if a deformation condition is met such that the pinhole deforms plastically. An empirical correlation is used to determine the FRR during fuel cell operation. The membrane properties are calculated as a function of FRR, which changes over time as the membrane degrades and forms feedback loop for mechanical and chemical degradation mechanisms.

3.1 Pinhole Model

During operation, a pinhole can form in the through-plane direction of the membrane and allow crossover gases to diffuse through the membrane and react at the opposite electrode. This provides transport pathway for crossover gases through the membrane in addition to permeation, which is a much slower process.[4, 54, 55] To account for the pinhole in the 1-D model, the pinhole is treated as an effective void fraction, which is used to modify the membrane properties. Assuming a cylindrical pinhole, the effective volume fraction of the pinhole is calculated as

$$\varepsilon_{\text{hole}} = \frac{\pi r_{\text{hole}}^2}{A} \tag{3.1}$$

where r_{hole} is the radius of the pinhole and A is the cross-sectional area of the membrane, which is taken to be 50 cm².[56] The membrane properties are modified by using a Bruggeman correction, assuming the effective volume fraction is equal to $1 - \varepsilon_{\text{hole}}$. The absolute permeability of the pinhole is taken to be $(\varepsilon_{hole}\ r_{hole}^2)/8$.[50]

In addition, the pinhole provides a direct pathway for gas transport through the membrane. This is modeled by extending the Stefan-Maxwell equations into the membrane domain. Flux through the pinhole is characterized by the pinhole radius and unit tortuosity. Transport through the pinhole is assumed to dominate the gas crossover compared to gas permeation through the membrane as given by Equation (2.17). To check this assumption, simulations were run to calculate gas crossover as a function of pinhole size. The results in Figure 3-1 show that this is a reasonable assumption except for very small pinholes ($r_{hole} < 100\ \mu$m). The hydrogen gas crossover through the pinhole is initially smaller than the gas crossover calculated by permeation ($r_{hole} = 0$), but it increases exponentially after this initial decrease. The oxygen gas crossover increases with pinhole size in all cases.

a) b)

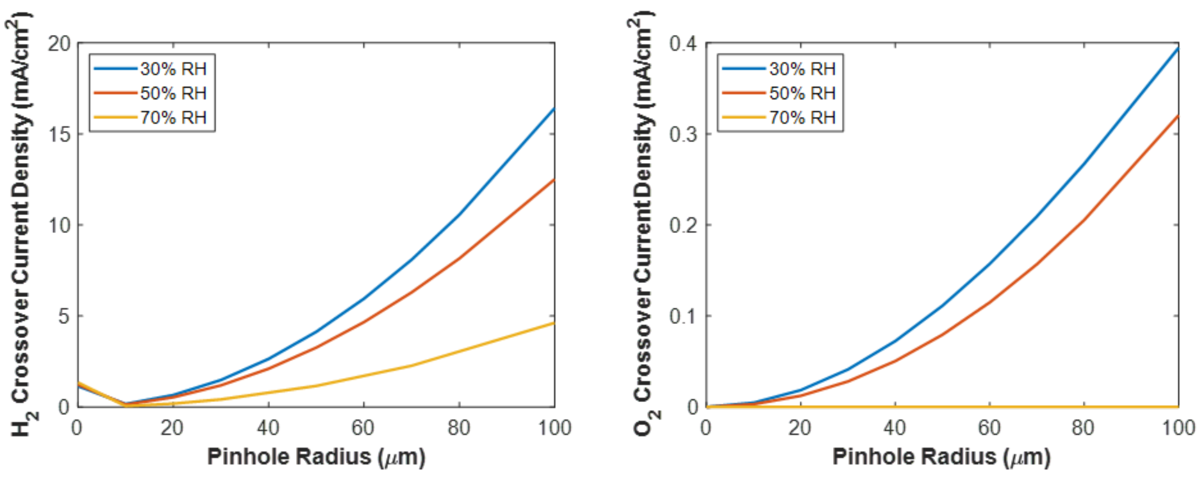


Figure 3-1: Gas crossover as a function of pinhole size. Simulations were run at 80°C, 1 bar, 0.65 V, and air/feed stoichiometry 1.2/2.

3.2 Membrane Mechanics Model

The mechanical model presented here builds on previous work by Weber and Kusoglu, who investigated the effects of gas crossover and membrane pinhole effects,[13] and developed a 0D mechanical model for pinhole growth in PEMFCs under RH cycling conditions.[12] In comparison, this model fully couples the fuel-cell performance and membrane mechanical models. The membrane mechanical model calculates the stresses acting on the membrane due to changes in RH. Several assumptions are made to simplify the expressions for mechanical stresses. In this model, an idealized circular pinhole is assumed to be already present in the membrane, as illustrated in Figure 3-2. The model calculates the elastic and plastic stresses that occur in the membrane under uniform, biaxial loading conditions. Swelling of the membrane is assumed to be isotropic and mechanical properties describing the elastic-plastic behavior of the membrane are also assumed to be isotropic. The isotropy assumptions need to be revisited for reinforced membranes that exhibit anisotropy in both swelling and mechanical properties,[57] and necessitates a much more complex mechanical model due to the existence of the reinforcement layer, which is beyond the scope of the current study. While such anisotropy impacts the stress distribution, its coupling to transport could still be captured by the current modeling frame.

The strain-stress constitutive relations for the elastic response are written using the generalized Hooke's law and assuming biaxial stress and biaxial strain. The equations simplify to

$$\sigma_x = \frac{E}{(1+\nu)(1-2\nu)}\left(\varepsilon_x^{el} + \nu\varepsilon_z^{el}\right) \tag{3.2}$$

$$\sigma_z = \frac{E}{(1+\nu)(1-2\nu)}\left(2\nu\varepsilon_x^{el} + (1-\nu)\varepsilon_z^{el}\right) \tag{3.3}$$

where σ_x and σ_z are stresses in the x- and z- directions, ε_x^{el} and ε_z^{el} are elastic strains in the x- and z- directions, E is Young's modulus, and ν is Poisson's ratio. The stress in the z-direction is assumed to be equal to a constant compressive stress $\sigma_z = -p$, assuming a spring-loaded fuel-cell construction forcing the membrane to remain in place; see Figure 3-2.

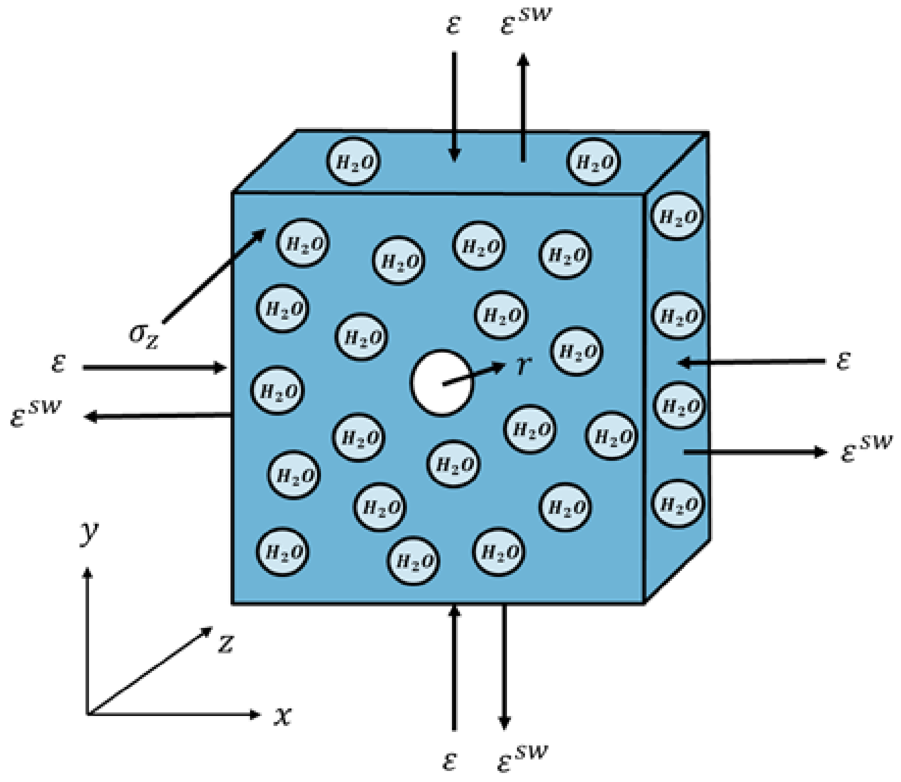


Figure 3-2: Schematic of swelling strain resultant mechanical (elastic and plastic) strain in the membrane.

Plastic deformation occurs when the equivalent stress, $\overline{\sigma}_e$, reaches the membrane's yield strength, σ_Y. In the absence of shear stresses, the following condition must always be satisfied:

$$\overline{\sigma}_e = |\sigma_x - \sigma_z| \leq \sigma_Y \tag{3.4}$$

The relationship between the plastic strain and stress in the in-plane direction is

$$d\varepsilon_x^{pl} = -d\varepsilon_x^{el} - d\varepsilon_x^{sw} = -\frac{1-\nu}{E} d\sigma_x - d\varepsilon_x^{sw} \tag{3.5}$$

where $d\varepsilon_x^{pl}$, $d\varepsilon_x^{el}$, and $d\varepsilon^{sw}$ are the incremental plastic, elastic, and swelling strains in the x-direction, respectively, and $d\sigma_x$ is the incremental stress in the x-direction.

Furthermore, the membrane mechanical properties and stress-strain response change as a function of water content. The dependence of Young's modulus and yield strength of the membrane on water uptake can be characterized using scaling laws fit to experimental data[58, 59] as shown in Figure 3-3 and described in Kusoglu and Weber (2014).[12] The stress-strain data are implemented into the model using the constitutive models developed by Kusoglu *et al.*[58, 60] When the membrane is fully hydrated, the Young's modulus and yield strength of the membrane are reduced to approximately 25 and 40% of the dry polymer values, respectively.

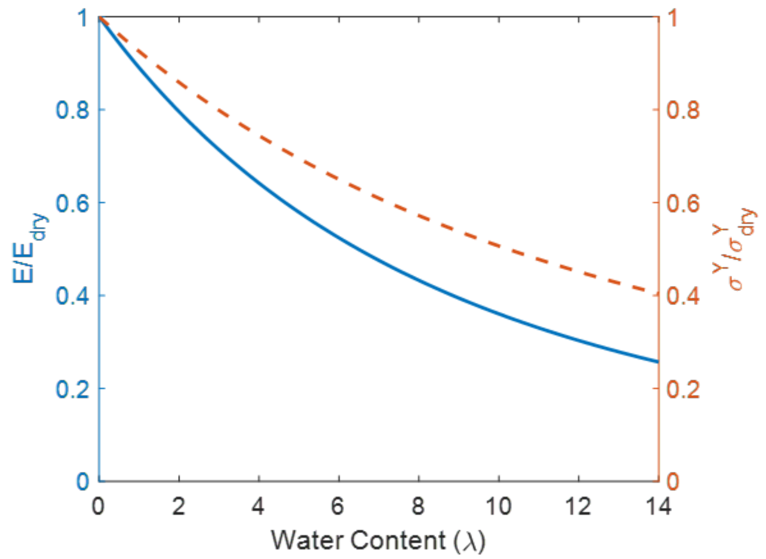


Figure 3-3: Membrane mechanical properties as a function of water content, reproduced from Kusoglu and Weber (2014).[12]

Finally, the change in the radius of the pinhole is calculated from the equivalent stress and equivalent plastic strain as

$$\frac{dr}{r} = c \exp\left(\frac{3}{2}\frac{\sigma_m}{\bar{\sigma}_e}\right) d\bar{\varepsilon}^{\mathrm{pl}} \tag{3.6}$$

where σ_m and $\bar{\sigma}_e$ are the far-field mean stress and equivalent stress, $\bar{\varepsilon}^{\mathrm{pl}}$ is the equivalent plastic strain associated with plastic deformation, and c is a number in the order of unity relating the void radius to remote strains, and its value as well as overall expression depends on model assumptions and void shape. In this case c is taken to be 0.283.[12] The membrane mechanical properties are listed in Table 3-1.

Table 3-1: Membrane Mechanical Properties [12]

Property		Units	Value
Young's modulus of dry polymer	E_{dry}	MPa	250
Yield strength of dry polymer	σ^Y_{dry}	MPa	7.5
Hardening exponent	h		2.2
Scaling exponent for Young's modulus	m		3.6
Scaling exponent for yield strength	p		2.4
Coefficient for radius growth	c		0.283

3.3 Empirical Degradation Model

Chemical degradation in PEMs is caused by peroxide formation and radical generation at the electrodes due to the natural ORR and especially crossover gases.[14] Hydroxyl radicals are generated from disproportionation of hydrogen peroxide, which then attack the ionomer and cause HF to be released, which can be measured in the fuel cell effluent water.[61, 62] Here, an empirical model for chemical degradation is used to couple the effects of chemical and mechanical degradation. The FRR is assumed to be directly proportional to the generation of hydrogen peroxide via the two-electron ORR. This FRR value is then used to adjust the membrane mechanical properties, which change over time as the membrane degrades. A full micro-kinetic model for chemical degradation of the membrane is described in Chapter 5.

Chemical degradation is accounted for through electrochemical generation of hydrogen peroxide. In the model, the rate of generation of peroxide can be calculated as a function of electrocatalytic surface area and oxygen concentrations. The kinetic equation for generation of hydrogen peroxide is taken as

$$\mathbf{i}_{\mathrm{ORR2}} = -i_{0,\mathrm{ORR}_{2e^-}} \frac{p_{O_2}}{p_{O_2}^{\mathrm{ref}}} \exp\left(-\frac{\alpha_c F}{RT} \left(\Phi_1 - \Phi_2 - \mathrm{U}_0^{\mathrm{ORR}_{2e^-}} \right) \right) \tag{3.7}$$

where $U_0^{ORR_{2e^-}}$ = 0.695 V versus SHE. The effectiveness factor for the two-electron oxygen reduction reaction is assumed to be the same as that for the four-electron oxygen reduction reaction. Empirical correlations relate hydrogen peroxide generation to FRR as found in Kundu *et al.*[63]

$$\frac{dn_{F^-}}{dt} = k\boldsymbol{i}_{\mathrm{ORR2}} \tag{3.8}$$

where n_{F^-} is the number of fluoride ions in moles, and the rate constant k = 4.0×10^{-7} mol/cm^2. The rate of diffusion of fluoride ions through the fuel cell is calculated using Fick's law, with a diffusion coefficient of 4.2×10^{-9} cm^2/s in the GDLs[63] and 2×10^{-10} cm^2/s in the CLs and membrane.[64] The FRR is the sum of the outward flux of fluoride ions out of the fuel cell into the gas channels. The cumulative FRR is a measure of chemical degradation as it quantifies the total amount of fluoride that has been released from the polymer over the course of fuel cell operation.

To couple the effects of chemical and mechanical degradation, the membrane modulus and membrane thickness are calculated as a function of FRR. An empirical correlation between FRR and Young's modulus of degraded membrane, E, was determined from experimental data based on *ex-situ* measurements (see Figure 3-4).[65] The effect of the cumulative FRR on the Young's modulus of the membrane is incorporated into the model based on a correction factor, E_{corr}

$$E_{corr} = (1 - (5.739 \times 10^{-3})\mathrm{FRR})E \tag{3.9}$$

where FRR is the cumulative fluoride emission in μmol and

$$E = (4 - 0.01T)\phi_p^m E_{\text{dry}} \tag{3.10}$$

where ϕ_p is the volume fraction of dry polymer in the membrane, m is an exponential scaling factor, T is temperature in K, and E_{dry} is the Young's modulus of the dry polymer. This expression is used as a scaling factor (E/E_0) to correct the Young's modulus of the dry polymer for degradation effects in the mechanical model and serves as one of the coupling physics between the models. An empirical correlation between FRR and membrane thickness was determined from experimental data in [66] and adjusted for the initial membrane thickness $L_{M,0}$ given the equivalent weight and fluoride content of the ionomer,

$$L_{\text{M}} = -2.2 \times 10^{-4} \log(FRR) + L_{M,0} \tag{3.11}$$

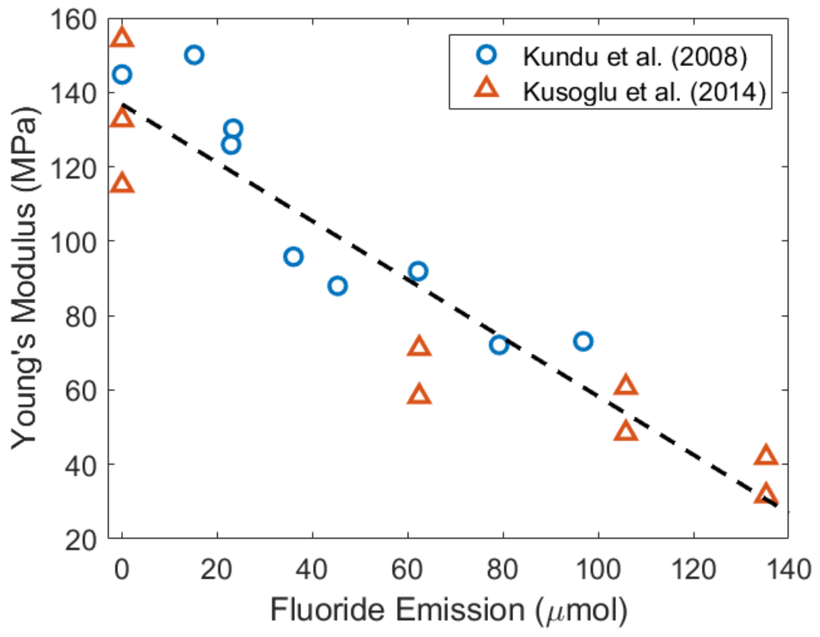


Figure 3-4: Linear fit of Young's modulus and fluoride emission.[65, 67]

3.4 Model Coupling and Solution

The fuel-cell performance model and membrane mechanical model are both run in MATLAB (see Appendix B for codes used). The fuel-cell operating conditions such as RH, temperature, and pressure are inputs to both models. To initialize the simulation, certain operating parameters such as temperature, pressure, feed stoichiometry, air stoichiometry, membrane properties, pinhole radius size, *etc*. must be specified. These parameters are used to calculate the initial condition for the transient simulation by solving the fuel-cell performance model at steady-state conditions. Correspondingly, the membrane mechanical model is solved to calculate the initial mechanical properties including Young's modulus and yield strength, which are a function of the initial water content λ_0 and degradation rate. See Figure 3-5 for a flowchart of the model solution method.

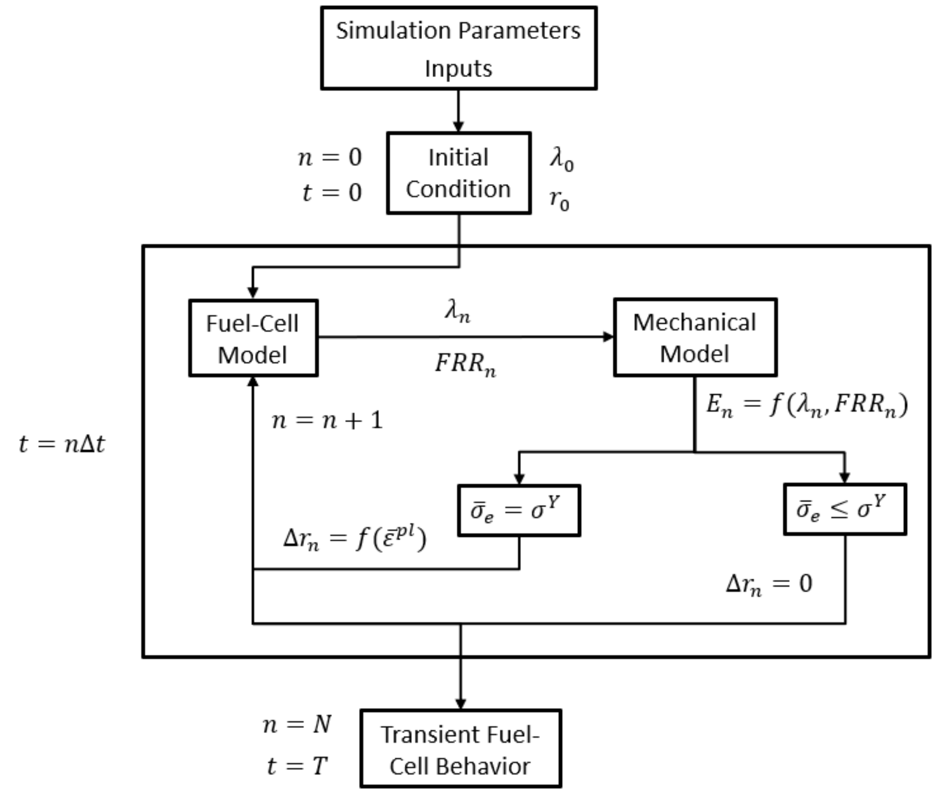

Simulation Parameters Inputs
$n = 0$
$t = 0$
Initial Condition
λ_0
r_0
Fuel-Cell Model
λ_n
FRR_n
Mechanical Model
$E_n = f(\lambda_n, FRR_n)$
$n = n + 1$
$t = n\Delta t$
$\bar{\sigma}_e = \sigma^Y$
$\bar{\sigma}_e \leq \sigma^Y$
$\Delta r_n = f(\bar{\varepsilon}^{pl})$
$\Delta r_n = 0$
$n = N$
$t = T$
Transient Fuel-Cell Behavior

Figure 3-5: Flowchart for coupling of fuel cell performance model and membrane mechanical model.

The performance model is solved to find the average activity for water in the membrane for the given initial conditions. The water-content value from the fuel-cell model is used to calculate the polymer volume fraction in the mechanical model. In the mechanical model, the membrane mechanical properties are calculated as a function of water content and the stress-strain relations are solved to determine if plastic deformation of the pinhole occurs. If the swelling strain causes a stress that meets the plastic deformation criterion, a new pinhole radius is calculated, and an updated volume fraction associated with that pinhole radius is used in the next time step for the fuel-cell performance model.

To solve the mechanical model, first the polymer volume fraction is calculated from the membrane water content, which is a known value calculated by the fuel-cell performance model. Then, the mechanical properties and swelling strains are calculated as a function of that water content. Next, the yield criterion, Equation (3.4), is evaluated to determine the magnitude of elastic and plastic strain. If the equivalent stress is equal to the yield strength, then the polymer deforms plastically, and the change in pinhole radius is determined from the calculated plastic strain.

Next, the simulation moves forward one time-step. The fuel-cell performance model calculates changes in fuel cell performance including gas crossover, water content, and FRR. The water content and FRR are then fed as inputs to the mechanical model. Based on the mechanical properties at each time step, the mechanical model determines if the membrane deforms plastically or not and the pinhole radius is updated. These steps are repeated until the final time is reached.

The governing equations are constructed using a finite-volume method approach, which enforces conservation of mass and energy. The system of equations is solved using a multidimensional Newton-Raphson technique developed by Newman.[52, 68] This technique is detailed in Appendix C of Newman and Thomas-Alyea.[68] To incorporate transient effects, a Crank-Nicolson approach is used to calculate the time derivatives in the mass- and energy-balance equations using an adaptive time-stepping method. The full set of equations and boundary conditions is listed in Appendix A.

In the simulation, a current density or a voltage must be specified for the cell. Inlet gas flow rates for hydrogen and air must be specified as a stoichiometric value. Temperature, pressure, and RH are specified at the anode and cathode gas channels. Simulations begin with an initial guess for all variables throughout the discretized fuel-cell domain and is then iterated until a converged solution is obtained.

3.5 Pinhole Effects

3.5.1 Nonlinear Behavior in Pinhole Simulation Results

Model simulations under varying pinhole sizes revealed several nonlinear trends in fuel-cell behavior as shown in Figure 3-6. The catalyst layers and membrane are discretized using 41 mesh points and the gas diffusion layer are discretized using 21 mesh points.

The overpotential for the hydrogen-oxidation reaction, which drives conversion of hydrogen gas to protons, decreases with pinhole size until about a pinhole radius of 500 μm, and then begins to increase. The gas pressure decreases as the pinhole size increases, due to equilibration between the anode and cathode gas channels by gas crossover. However, the difference in pressure across the pinhole decreases with pinhole size, where it remains fairly constant at pinhole sizes above 100 μm. Consistent with the trends observed with overpotential, the temperature of the cell increases until 250 μm is reached, and then decreases as the pinhole grows. The initial temperature increase is due to the increase in overpotential for the ORR, which causes most of the heat generation in the cell in the cathode catalyst layer. However, as the anode and cathode gas channel concentrations equilibrate due to gas crossover, so does the temperature. Finally, the water-vapor mole fraction initially increases with pinhole size due to the increase in crossover hydrogen reacting at the cathode to generate water. The excess water begins to accumulate in the pinhole until the water starts to diffuse out of the cell through the cathode gas channel. The water concentration then begins to approach a flat profile across the entire fuel-cell sandwich as the pinhole keeps growing and the gas channels approach equilibrium.

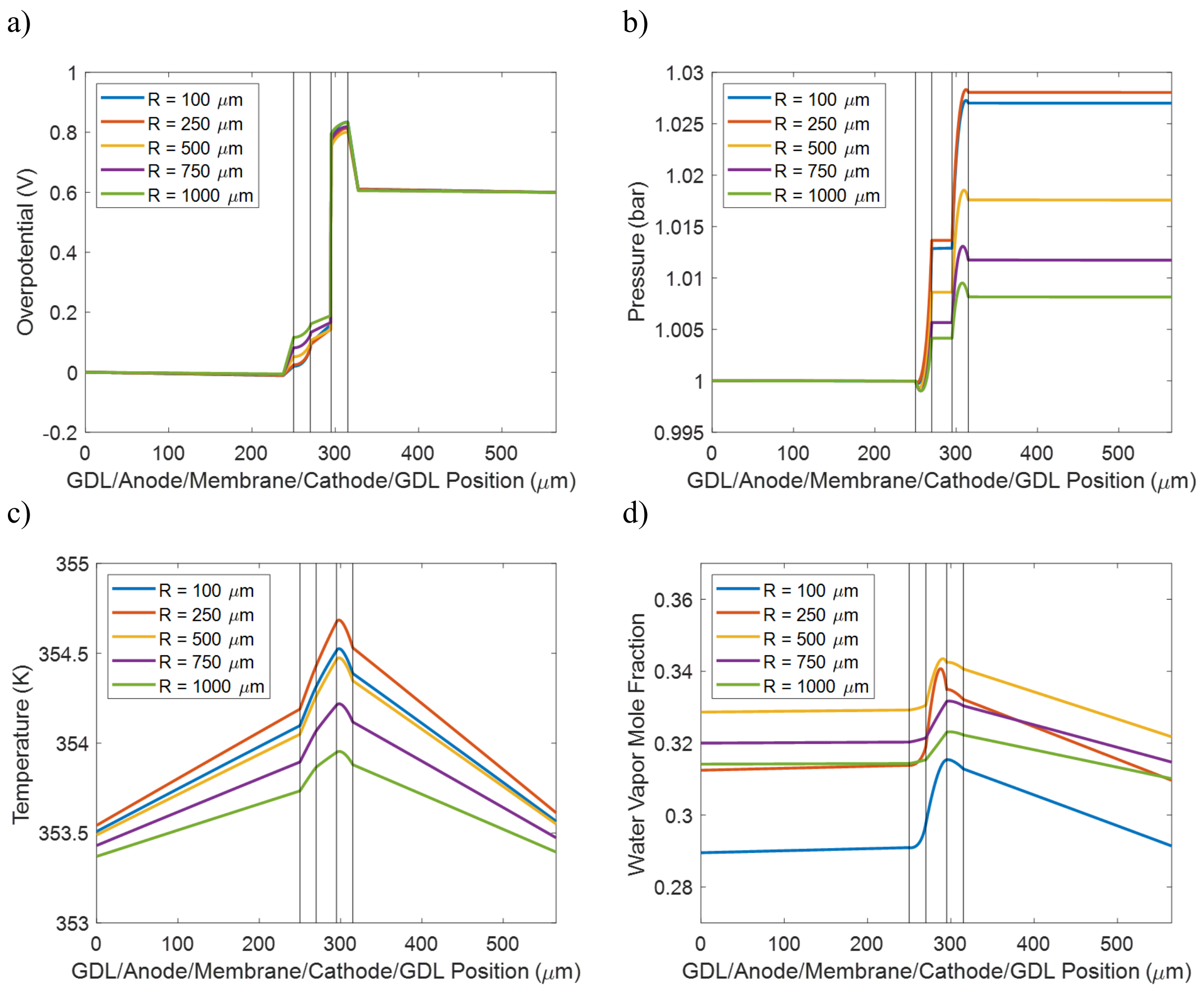


Figure 3-6: Simulation results for 25% RH at the anode and cathode and a cell potential of 0.6 V at various pinhole sizes, illustrating the nonlinear trends observed due to pinhole growth including a) overpotential for the hydrogen oxidation reaction, b) pressure, c) temperature, and d) water-vapor mole fraction.

3.5.2 Results for Gas Flux through the Pinhole

The simulation results in Figure 3-7 show the crossover gas fluxes at the mid-point of the membrane. Fluxes are defined as positive in the direction of anode to cathode and negative in the direction of cathode to anode. The hydrogen gas crossover makes up the majority of gas crossover through the pinhole, which corresponds to hydrogen's higher diffusivity and lower molecular weight compared to oxygen and nitrogen on the cathode side. Oxygen crossover increases steadily with increasing pinhole size. Nitrogen crossover increases and then peaks at around 250 μm pinhole radius and then decreases again so that there is zero net flux of nitrogen through the membrane. This due to a shift the equilibrium between the membrane-bound water and the water vapor where more water will remain the vapor phase.

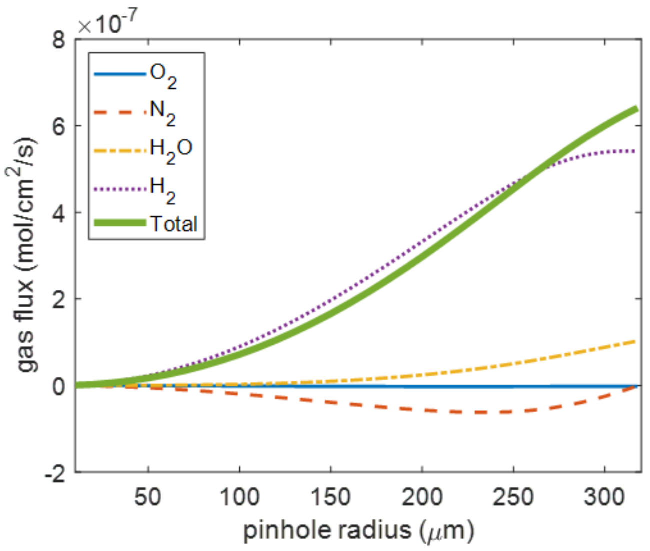


Figure 3-7: Gas fluxes at the center of the pinhole in the membrane at a voltage of 0.65 V and 25% RH at the anode and cathode.

3.5.3 Pinhole Effects on Cell Performance

Simulations were run at various pinhole sizes to analyze the impact of pinhole size on performance. The results in Figure 3-8 show that even for very small pinholes, fuel-cell performance is greatly decreased under typical operating conditions, which agrees with prior simulations.[13] The current density decreases with increasing pinhole radius due to reaction of crossover gases creating a mixed potential and changing membrane properties as a result of membrane degradation. At higher potentials, the cell fails at a smaller pinhole size because of the lower current. Both hydrogen and oxygen gas crossover increase with pinhole radius size and are higher at lower voltages, but crossover of hydrogen dominates. The fact that crossover is higher at lower voltages stems from the nonlinearity of the phenomena and particularly the water management with the subsaturated feeds (see Figure 3-6).

FRR increases with increasing pinhole size until about 150 to 250 μm pinhole radius, then it starts decreasing due to the mixed potentials resulting from the high gas crossover. As the pinhole size increases, the total gas flow through the pinhole increases and ionic transport decreases, since a larger hole allows more gas to flow through. This is confirmed by analysis of the gas fluxes through the pinhole (see Figure 3-7). Interestingly, the higher crossover does have a beneficial effect of keeping the membrane better hydrated due to the crossover reactions providing additional water generation.

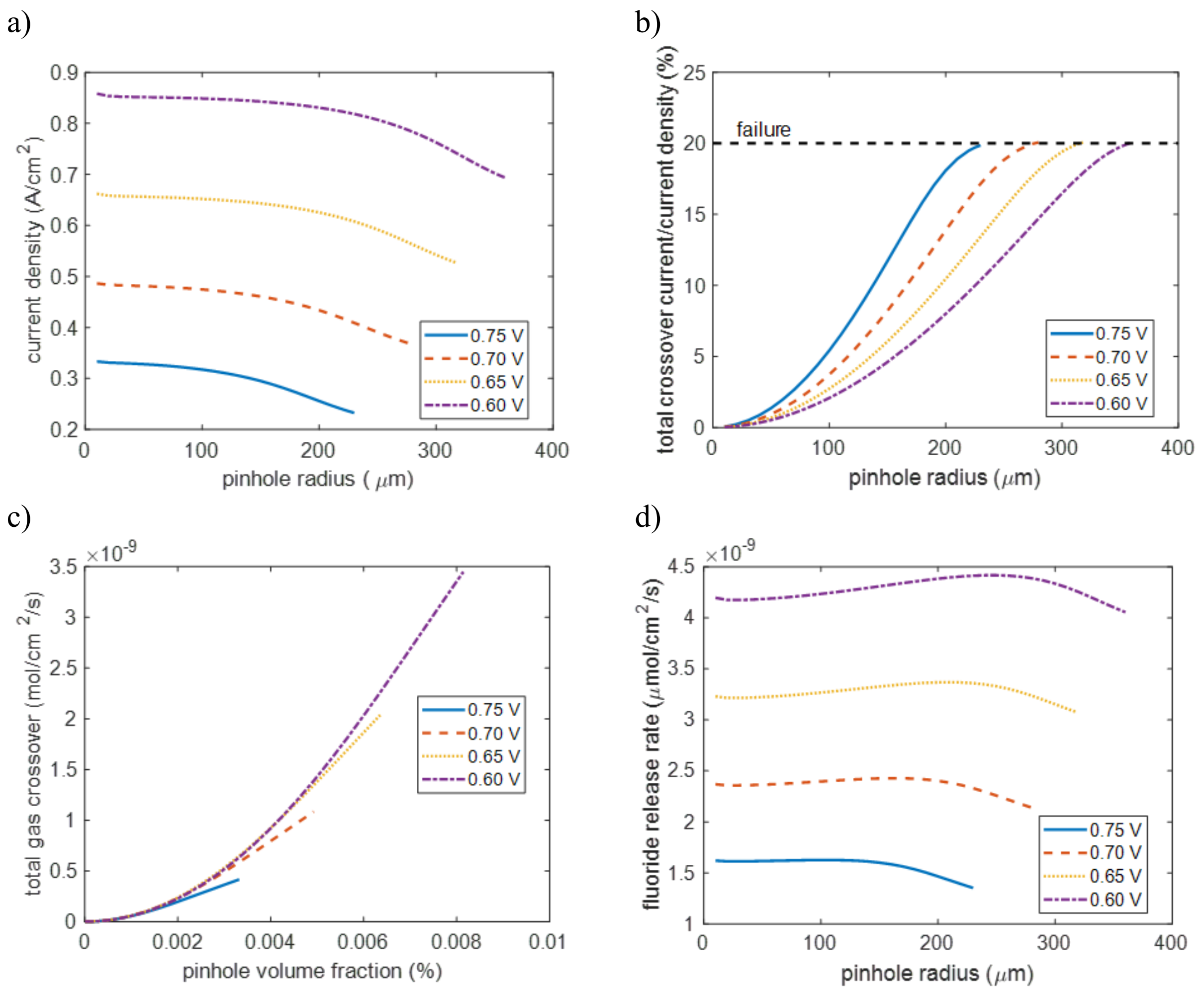


Figure 3-8: a) Current density as a function of pinhole size at several voltage values. b) Ratio of crossover current density to current density. c) Total gas flow through the membrane in the center of the pinhole. d) Corresponding FRR into the gas channel, which is a summation of fluoride flux from the anode and cathode gas channels.

3.6 Mechanical Degradation in RH Cycling

To understand the impacts of mechanical degradation and pinhole growth, transient simulations were run over cycles of varying RH at a constant voltage of 0.7 V. The RH cycles begin at an initial value of 25% RH at both the anode and cathode and are increased to 55, 65, and 75% RH to analyze the impact of swelling strain at different water-uptake rates. These operating conditions vary throughout fuel-cell lifetime due to environmental conditions and varying power output needs due to acceleration and deceleration. The characteristic time constants for fuel-cell transient phenomena are provided by Wang and Wang,[69] and the model results herein agree with their calculated time constant values, with water uptake having the largest time constant. Double-layer charging can be ignored due to its fast time constant. The time constants for water uptake (τ_M), gas diffusion (τ_g), and double-layer charging (τ_{dl}) are given by

$$\tau_m = \frac{(\rho_M L_M \Delta\lambda)/EW}{i/2F} \tag{3.12}$$

$$\tau_g = \frac{L_{GDL}^2}{D_g^{\text{eff}}} \tag{3.13}$$

$$\tau_{dl} = L_{CL} a C \left(\frac{1}{\kappa} + \frac{1}{\sigma}\right) \tag{3.14}$$

where L_M, L_{CL}, and L_{GDL} are the thicknesses of the membrane, catalyst layers, and gas diffusion layers, respectively (see Table 2-2) , $\Delta\lambda$ is the change in water content, D_g^{eff} is the effective gas diffusivity, a is the electrode specific interfacial area, and C is the capacitance (typically 20 μF/cm^2).[69]

The fuel-cell performance model coupled with the membrane degradation model demonstrates that the plastic deformation coincides with the largest rate of change in RH due to increased swelling strain. The trends on membrane behavior are in agreement with the results from 2-D computational mechanics models of PEMs.[11, 60]

The plastic deformation occurring in response to the change in RH, along with water production and transport during operation, impact the membrane water content. During the first hydration cycle, the membrane water content increases and the swelling strain causes the pinhole radius to increase. During this plastic deformation the membrane undergoes strain hardening and the yield strength of the membrane increases. As a result, the magnitude of the plastic deformation during the second cycle is less than the first cycle. At a low enough water content, no plastic deformation will occur because the swelling strain is not great enough to cause the equivalent stress to equal the yield strength. The results in Figure 3-9 show that for a change in RH less than 30% no plastic deformation occurs. The simulation results demonstrate a linearly increasing cumulative FRR due to the constant rate of electrochemical generation of hydrogen peroxide at a constant voltage. The hydrogen crossover current density decreases at high RH values due to the increased membrane conductivity and current density, which results from more of the inlet hydrogen gas is being reacted. Furthermore, the hydrogen crossover current increases at lower RH values as the magnitude of the RH cycles increases due to the increase in pinhole radius, thus coupling the increase in both chemical and mechanical degradation rates. Additionally, the FRR increases with increasing magnitude of RH cycles due to the increase in the gas crossover through the membrane.

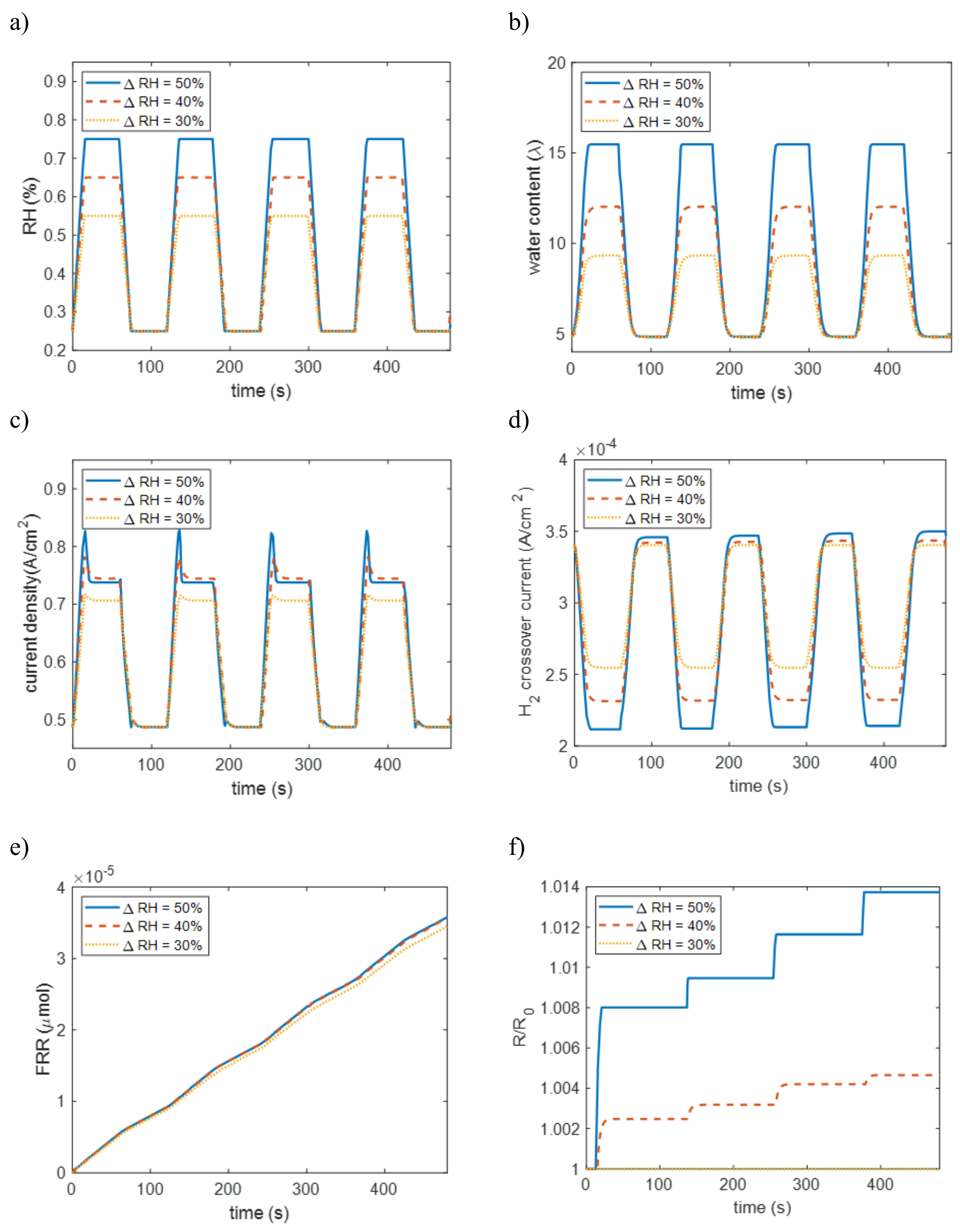


Figure 3-9: Fuel-cell and mechanical coupled model simulation results run under RH cycling at 0.7 V. a) Specified RH cycle profiles, b) water-content, c) current density, d) hydrogen gas crossover current density, e) cumulative FRR, and f) the ratio of pinhole radius (R) over initial pinhole radius (R_0) during RH cycling.

3.7 Chemical Degradation during Voltage Cycling

Similar to the RH cycling simulations, several voltage cycling simulations were run to illustrate the effects voltage cycling on chemical degradation rates. The voltage cycles begin at an initial value of 0.6 V and approach the OCV as the voltage increases to 0.7, 0.8, 0.9, and 1.0 V. Figure 3-10 shows the model results for voltage cycling at a constant inlet gas flow rate at both the anode and cathode feed channels. The simulation results in Figure 3-10 show that changes in potential during a potential cycle do not lead to plastic deformation in the membrane. This is a result of a small change in water content in the membrane, and therefore a correspondingly small swelling strain. The water content decreases with increasing voltage, due to the lower current density and therefore lower water generation rate at the cathode. The current density decreases with increasing voltage and approaches zero current as the voltage approaches OCV. Additionally, the FRR decreases with increasing voltage due to a lower overpotential for the two-electron oxygen reduction reaction. The hydrogen crossover current density increases with voltage as a result of higher overpotential driving the hydrogen oxidation reaction. The relationship between hydrogen crossover current density and OCV is shown in Figure 5-2.

Additional simulations were run with alternating cell voltage cycles from 0.6 to 0.9 V at a constant RH of 30% at the anode and cathode. The results in Figure 3-11 show that the presence of a pinhole allows for improved hydration of the membrane due to accumulation of water vapor in the pinhole; subsequently, the conductivity of the membrane increases. However, as the pinhole grows the pressure differential and concentration gradients from anode to cathode equilibrate and hydration of the membrane decreases. While there is a small decrease in current density due to an increase in pinhole size, a large increase in gas crossover current density is observed with larger pinhole sizes, thus reflecting an increase in chemical degradation rate, which is also shown as an increase in the FRR.

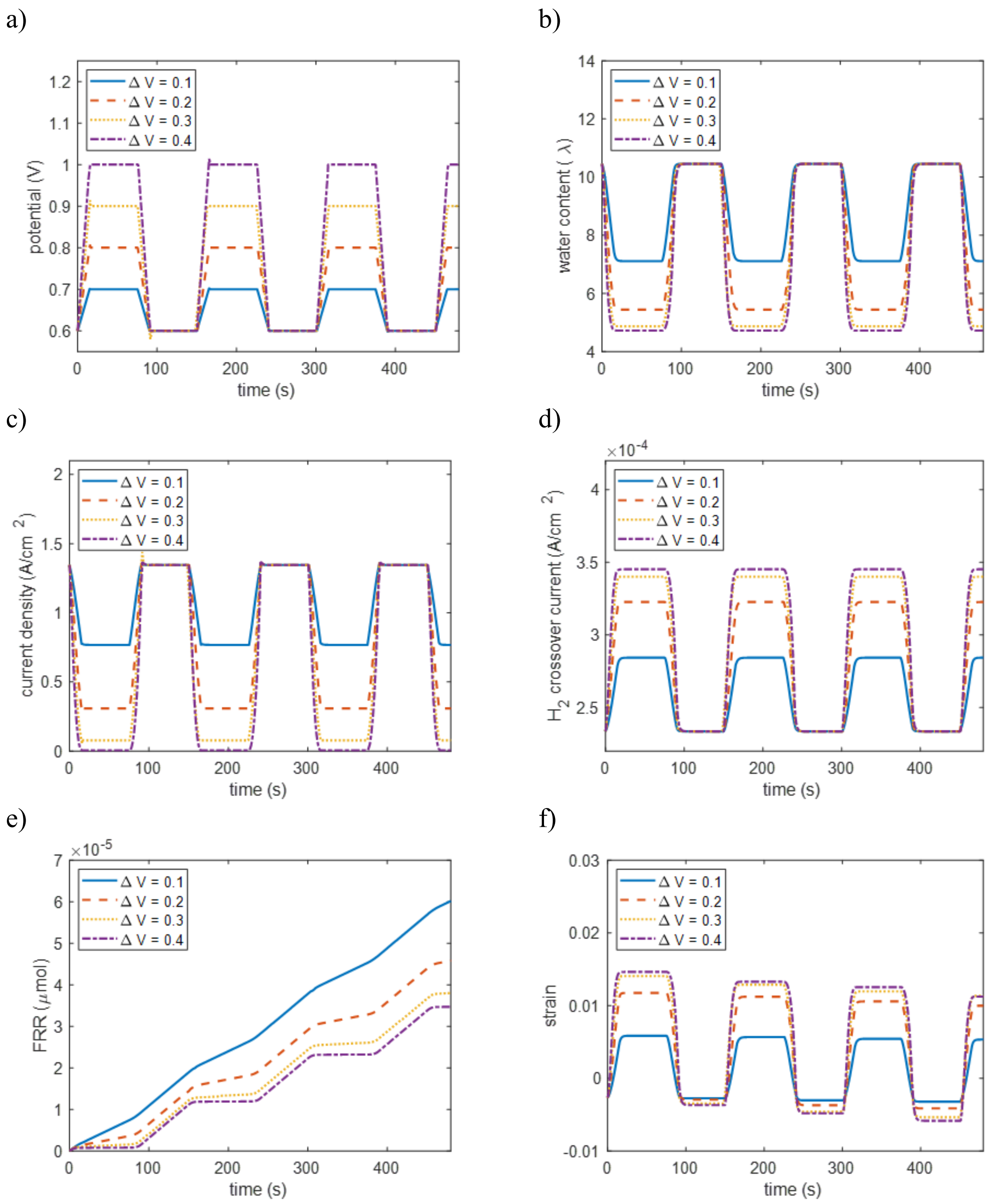


Figure 3-10: Fuel-cell and mechanical coupled model simulation results run under potential-cycling at constant 30% RH at the anode and cathode and constant inlet flow rates. The feed flow rate is 8.4×10^{-6} mol/cm^2/s and the air flow rate is 3.3×10^{-5} mol/cm^2/s. a) Potential cycle specified, b) water content, c) current density, d) hydrogen gas crossover current density, e) cumulative FRR, and f) strain in the membrane during voltage cycling.

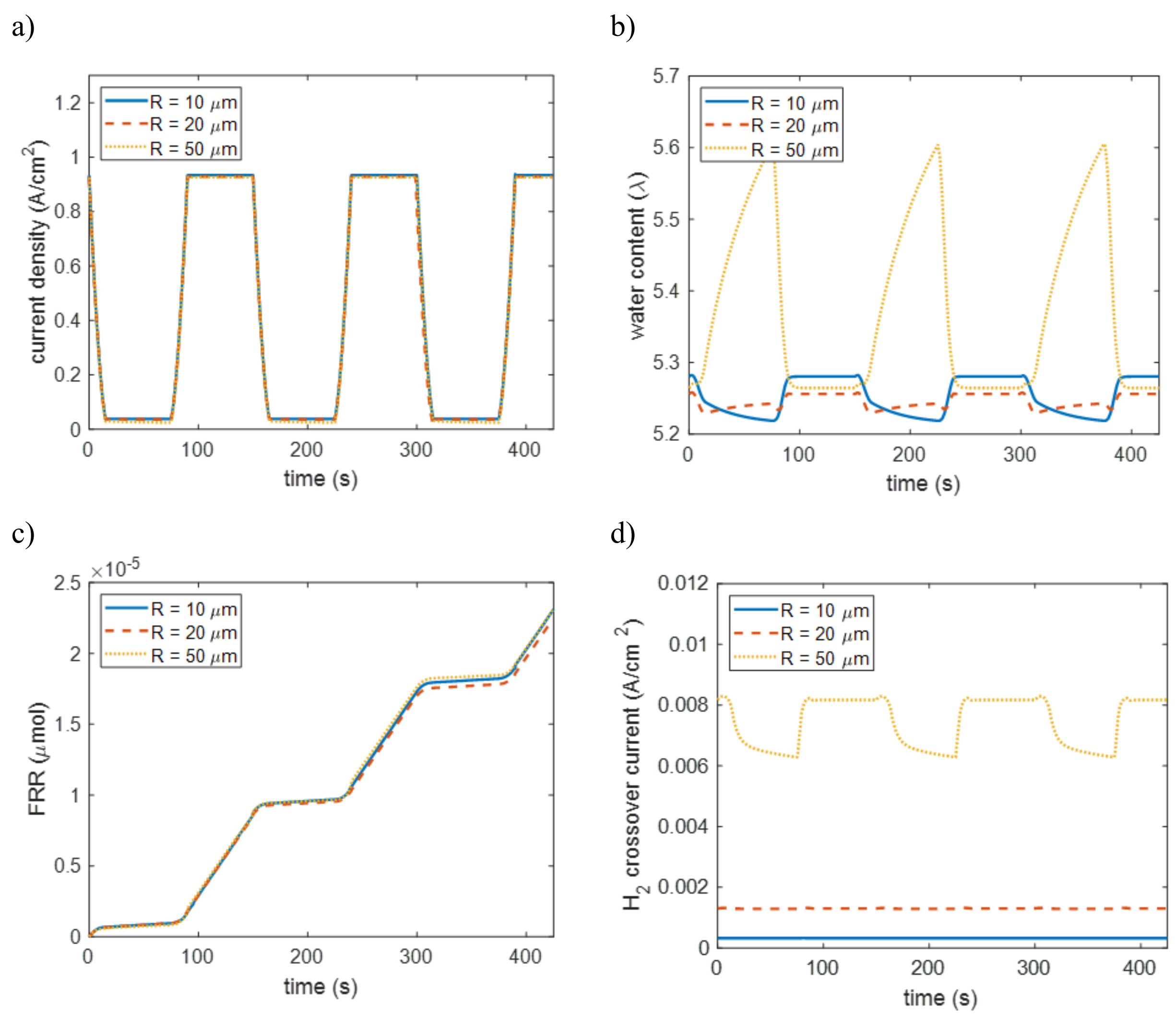


Figure 3-11: Fuel-cell and mechanical coupled model simulation results run under voltage cycling conditions from 0.6 V to 0.9 V at 30% RH at the anode and cathode with varying pinhole sizes. a) Current density, b) water content, c) cumulative FRR, and d) gas crossover current density resulting from voltage cycling.

3.8 Effects of Mechanical Properties

3.8.1 Young's Modulus

Chemical degradation of the membrane leads to a change in mechanical properties and therefore results in acceleration of degradation until the fuel cell fails. As the fuel cell runs, the cumulative FRR increases. With current PEMs, the FRR is quite small, such that it takes several hundred hours for measurable quantities of fluoride to be observed in the gas channel effluent.[15, 19, 61, 63, 70, 71]

To explore the coupled nature of performance and degradation, simulations were run to demonstrate the effects of changing material properties over time as a result of coupled mechanical and chemical degradation. As shown in Figure 3-3, the Young's modulus of the membrane decreases with increasing FRR as a result of a changes in the membrane structure due to chemical degradation. Results in Figure 3-12 demonstrate the effect on the degradation rate of the membrane under RH-cycling conditions. As the Young's modulus decreases, the membrane generates lower stresses for a given strain and therefore deforms elastically over a larger range of strains. The simulation results indicate that the pinhole will no longer deform plastically once a cumulative FRR over 100 μmol has been reached.

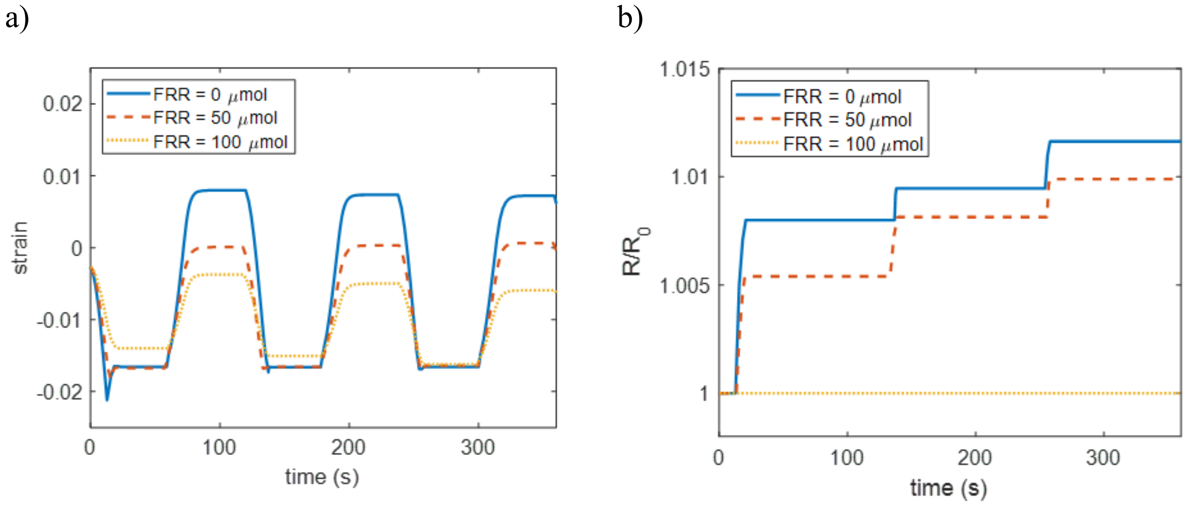


Figure 3-12: Fuel-cell and mechanical coupled model simulation results run at 0.7 V under RH cycling from 25% to 75% RH at the anode and cathode. a) Strain in the in-plane direction at various cumulative FRR values and corresponding change in Young's modulus. b) Ratio of pinhole radius over initial pinhole radius at various cumulative FRR values and corresponding change in Young's modulus.

3.8.2 Yield Strength

Additionally, as the membrane becomes thinner, due to non-localized chemical degradation, one would expect that the yield strength of the membrane would also change; the strain decreases with an increase in yield strength. Simulation results for varying polymer yield strength values and pinhole deformation rates are shown in Figure 3-13. In addition, pinhole deformation during RH-cycling increases at a higher rate for a membrane with a lower yield strength, all other things being equal. Hence, the coupled transport/mechanics model developed herein demonstrates how implementing the effect of chemical degradation on mechanical properties yield complex interplays with highly nonlinear and non-monotonic trends of defect behavior during operation.

Figure 3-13 shows the effects of membrane thinning on membrane mechanical and chemical degradation. The membrane plastic deformation increases and the gas crossover and FRR both increase with decreasing membrane thickness. Therefore, when membrane thinning occurs, chemical and mechanical degradation both increase and accelerate the overall degradation.

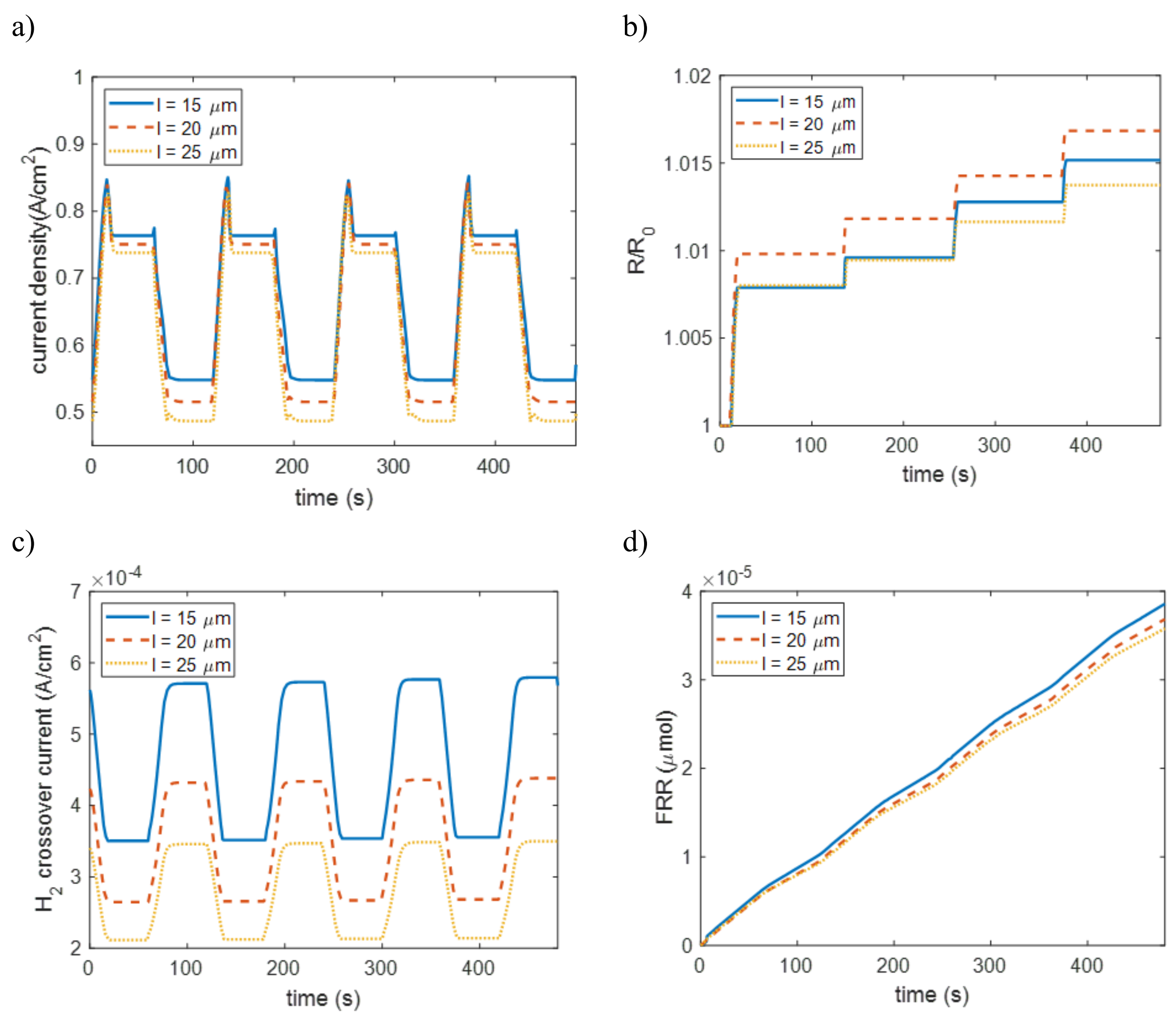


Figure 3-13: Change in membrane thickness effect on performance under RH cycling. a) Decrease in current density. b) Increase in plastic deformation. c) Increase in gas crossover. d) Increase in FRR.

Chapter 4 – Membrane Degradation with Multiphase Phenomena

At higher RH, multiphase phenomena must be considered as the water vapor may condense to form liquid water. The formation of liquid water in the cell causes flooding, which reduces cell performance. In this section, a multiphase model for water in PEMFCs developed by Weber and Newman[44-47] is coupled with the membrane mechanical model and empirical chemical degradation model in Chapter 3 in order to analyze the effects of high humidity on membrane degradation. Large changes in RH cause swelling stress in the membrane and lead to pinhole growth, as demonstrated in Chapter 3. In addition, at high humidity values, the water vapor may condense inside the pinhole and restrict the additional transport pathway for crossover gasses. In the catalyst layer and other porous media, flooding reduces the availability of catalyst sites and oxygen transport, which not only causes a decrease in cell performance, but also leads to a decrease in peroxide generation and FRR. Thus, there are multiple facets that need to be considered and it is not clear a priori which ones dominate.

4.1 Membrane Structure

Schroeder's paradox describes the difference in solvent uptake observed in solid polymers, such as gels or membranes, when in contact with a saturated vapor versus a saturated liquid. Figure 4-1 shows how Schroeder's paradox is observed in the water uptake of PFSA membranes. This paradox seemingly breaks phase equilibrium, as both the saturated vapor and saturated liquid have the same chemical potential. However, this phenomenon can be explained by the change in morphology of the ionomer nanostructure as it swells.[44, 51] Since the PFSA is composed of a hydrophobic backbone with hydrophilic ionic end groups, the polymer separates into a non-conducting hydrophobic phase and a hydrophilic ion-conducting phase. As water is absorbed into the membrane, the hydrophilic domains grow and eventually form clusters and connecting channels.[51]

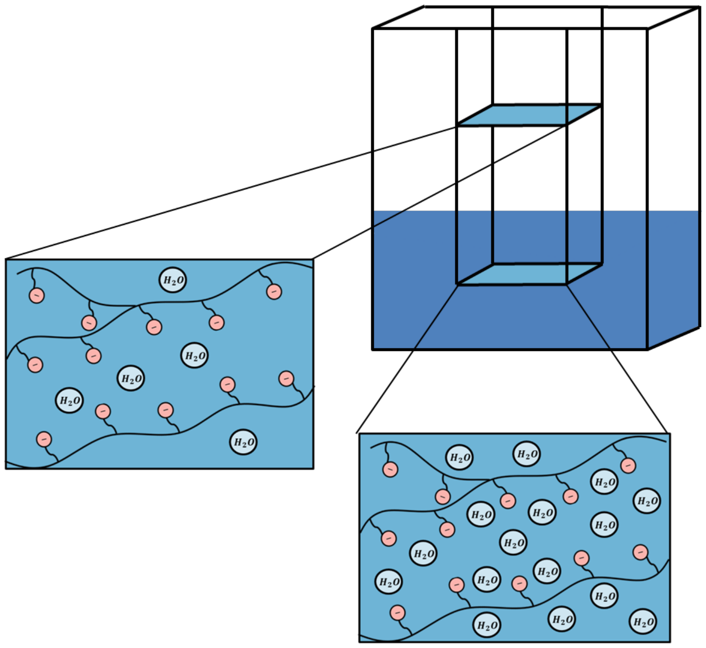


Figure 4-1: Illustration of Schroeder's paradox in PEMs. Water content of a membrane in contact with saturated vapor is less than the water content of a membrane in contact with liquid.

4.2 Multiphase Phenomena

4.2.1 Porous-Medium Model

To account for the presence of liquid water in the fuel cell, a porous medium model is used to determine the fraction of pores that are saturated by liquid water. The porous medium is characterized by pore size distribution (PSD) of hydrophobic and hydrophilic domains. If the fraction of hydrophilic pores is less than 15% or greater than 85%, a single PSD is used and a composite angle is calculated,

$$\theta_c = \arccos(f_{HI} \cos\theta_{HI} + (1 - f_{HI}) \cos\theta_{HO}) \tag{4.1}$$

where f_{HI} is the fraction of hydrophilic pores, θ_{HI} and θ_{HO} are the hydrophilic and hydrophobic contact angles, respectively. Otherwise the porous medium is assumed to separate into two pore networks, one hydrophilic and the other hydrophobic, and a composite angle is calculated for each.

$$\theta_{c,HI} = \arccos(0.85 \cos\theta_{HI} + 0.15 \cos\theta_{HO}) \tag{4.2}$$
$$\theta_{c,HO} = \arccos(0.15 \cos\theta_{HI} + 0.85 \cos\theta_{HO}) \tag{4.3}$$

Once the critical angles have been determined, a critical radius value is are calculated for each. The critical radius $r_{c,h}$ of pore type h (either hydrophilic or hydrophobic) is defined as,

$$r_{c,h} = -\frac{2\gamma \cos\theta_h}{p_c} \tag{4.4}$$

where γ is the surface tension, θ_h is the contact angle of pore type h, and p_c is the capillary pressure, which is equal to the difference in the liquid pressure and gas pressure ($p_c = p_L - p_G$). The surface tension of water is shown in Table 2-1.

The saturation, S, is the volume fraction of pores that are filled with liquid water. The saturation is calculated by integrating over the pore size distribution and dividing by the total pore volume. The fraction of pores that are filled with liquid water can be calculated as,

$$S_h = \frac{f_1}{2}\left(1 + \vartheta_h \operatorname{erf}\left(\frac{\ln r_{c,h} - \ln r_{0,1}}{s_1\sqrt{2}}\right)\right) + \frac{f_2}{2}\left(1 + \vartheta_h \operatorname{erf}\left(\frac{\ln r_{c,h} - \ln r_{0,2}}{s_2\sqrt{2}}\right)\right) \tag{4.5}$$

where h is the type of pore (HI or HO), f_1, f_2, $r_{0,1}$, $r_{0,2}, s_1$ and s_2 are the fraction of the total distribution, the characteristic pore size, and the spread of pore size distributions 1 and 2, respectively, ϑ_h is defined as 1 for hydrophilic pores and −1 for hydrophobic pores, and erf is the error function.[47] The total saturation is calculated by

$$S = f_{HI} S_{HI} + (1 - f_{HI}) S_{HO} \tag{4.6}$$

The pore size distribution properties used in this model are listed in Table 4-1. For a porous-medium property Y that is a function of saturation, the overall value can be calculated in a similar manner,

$$Y = f_{HI}Y_{HI} + (1 - f_{HI})Y_{HO} \tag{4.7}$$

where Y_{HI} is the hydrophilic property value and Y_{HO} is the hydrophobic property value.

Table 4-1: Pore Size Distribution Properties[47]

Property		**Units**	**Gas Diffusion Layers**	**Catalyst Layers**	**Membrane**
Fraction of hydrophilic pores	f_{HI}		0.5	0.3	0
Hydrophilic contact angle	θ_{HI}	degrees	45	80	90.02
Hydrophobic contact angle	θ_{HO}	degrees	110	100	90.02
Characteristic pore size of distribution 1	$r_{0,1}$	μm	6	0.2	0.00125
Characteristic pore size of distribution 2	$r_{0,2}$	μm	0.7	0.05	0.00125
pore size distribution 1 width	s_1		0.6	1.2	0.3
pore size distribution 2 width	s_2		0.6	0.5	0.3
Fraction of pore size distribution 1	f_1		1	0.5	1

Once a porous medium has hydrated there is a certain amount of water that is difficult to remove, which is called the residual liquid saturation.[47] The residual liquid saturation is calculated using a fit from Monte-Carlo simulations,[47]

$$S_L^0 = -53202\varepsilon_0^5 + 17.062\varepsilon_0^4 - 21.706\varepsilon_0^3 + 13.692\varepsilon_0^2 - 4.816\varepsilon_0 + 0.9989 \tag{4.8}$$

From the residual liquid saturation, the residual gas saturation can be determined. The residual gas saturation is $S_G^0 = 1 - S_L^0$ if $S_L^0 \leq 0.15$ and $S_G^0 = 0.85$ if $S_L^0 > 0.15$. The tortuosity value used in the Bruggeman expression for adjusting material properties for porous materials is then modified,

$$\tau_G = [\varepsilon_0(S_G^0 - S)]^{-0.5} \tag{4.9}$$

where τ_G is the tortuosity of the void volume of the gas phase. The Knudsen radius used in Equation (2.3) for gas diffusion is recalculated using the pore size distribution properties and Equation (4.7),

$$r_{K,h} = f_1 r_{0,1} \exp\left(\frac{s_1^2}{2}\right)\left(\frac{1-\vartheta_h \operatorname{erf}\left(\frac{\ln r_{c,h} - r_{0,1}}{s_1\sqrt{2}} - \frac{s_1}{2}\right)}{1-\vartheta_h \operatorname{erf}\left(\frac{\ln r_{c,h} - r_{0,1}}{s_1\sqrt{2}}\right)}\right) + f_2 r_{0,2} \exp\left(\frac{s_2^2}{2}\right)\left(\frac{1-\vartheta_h \operatorname{erf}\left(\frac{\ln r_{c,h} - r_{0,2}}{s_2\sqrt{2}} - \frac{s_2}{2}\right)}{1-\vartheta_h \operatorname{erf}\left(\frac{\ln r_{c,h} - r_{0,2}}{s_1\sqrt{2}}\right)}\right) \tag{4.10}$$

$$r_K = f_{HI} r_{K,HI} + (1 - f_{HI}) r_{K,HO} \tag{4.11}$$

4.2.2 Liquid-Water Transport

When the membrane is in equilibrium with liquid water, the primary driving force for water transport is the water pressure gradient. The pressure-driven flux for liquid water can be written as,

$$\mathbf{N}_{w,L} = -\frac{k}{\bar{V}_w \mu} \nabla p_L \tag{4.12}$$

where k is the effective permeability, $\bar{V}_w$ is the molar volume of water, μ is the viscosity, and p_L is the liquid water pressure. The viscosity of water is listed in Table 2-1. The permeability is calculated as a function of saturation and using Equation (4.7) becomes

$$k_h = \frac{1}{2}\left(\frac{S - S_L^0}{1 - S_L^0}\right)^2 \left[\frac{f_1}{2}\left(1 + \vartheta_h \operatorname{erf}\left(\frac{\ln r_{c,h} - \ln r_{0,1}}{s_1\sqrt{2}} - s_1\sqrt{2}\right)\right) + \frac{f_2}{2}\left(1 + \vartheta_h \operatorname{erf}\left(\frac{\ln r_{0,h} - \ln r_{0,2}}{s_2\sqrt{2}} - s_2\sqrt{2}\right)\right)\right] \tag{4.13}$$

$$k = f_{HI} k_{HI} + (1 - f_{HI}) k_{HO} \tag{4.14}$$

At the gas-channel/gas-diffusion-layer interface the boundary condition for liquid water ensures that the liquid flux at the gas channel is zero until the liquid pressure is above the breakthrough pressure by using a hyperbolic step function[72]

$$N_{w,L} = k(p_L - p_{thru})[\tanh(p_L - p_{thru}) + 1] \tag{4.15}$$

where k = 0.1 and $p_{thru} = 1.02$ bar is the breakthrough pressure. Furthermore, the flux of liquid water in the membrane is assumed to be zero ($N_{w,L} = 0$).

4.2.3 Effects of Liquid Water in the Ionomer

When both the liquid- and vapor-transport modes occur, an additional equation is needed to relate the liquid pressure and the water chemical potential.

$$\nabla\mu_w|_V = \bar{V}_w \nabla p_L|_L \tag{4.16}$$

This equation represents the water liquid/vapor equilibrium in the system.

When the membrane is not fully equilibrated with vapor or liquid, the transport mechanisms between the liquid and gas phases are assumed to occur in parallel. Equations (2.15) and (2.16) are modified such that the current and water flux are treated as the sum of the two transport modes based on overall saturation,

$$\mathbf{i}_2 = S\left(-\kappa_L\nabla\Phi_2 - \frac{\kappa_L\xi_L}{F}\bar{V}_w\nabla p_{L,M}\right) + (1-S)\left(-\kappa_V\nabla\Phi_2 - \frac{\kappa_V\xi_V}{F}\nabla\mu_w\right) \tag{4.17}$$

$$\begin{aligned}\mathbf{N}_w = {} & S\left(-\frac{\kappa_L\xi_L}{F}\nabla\Phi_2 - \left(\alpha_L + \frac{\kappa_L\xi_L^2}{F^2}\right)\bar{V}_w\nabla p_{L,M}\right) \\ & + (1-S)\left(-\frac{\kappa_V\xi_V}{F}\nabla\Phi_2 - \left(\alpha_V + \frac{\kappa_V\xi_V^2}{F^2}\right)\nabla\mu_w\right)\end{aligned} \tag{4.18}$$

where $p_{L,M}$ is the liquid pressure in the membrane, κ_L, ξ_L, and α_L are the liquid equilibrated membrane conductivity, electroosmotic coefficient, and membrane transport coefficient, respectively. The liquid-equilibrated membrane properties are listed in Table 4-2. The overall water content in the membrane can be calculated as a function of saturation,

$$\lambda = (1-S)\lambda_V + \lambda_L S \tag{4.19}$$

where λ_V is the vapor-equilibrated water content determined by Equation (2.18), and λ_L is the liquid-equilibrated water content, which is taken to be a value of 22.

Additionally, liquid water in the catalyst layer is assumed to reduce the number of reactive sites available by forming a water film over the agglomerates, so Equations (2.9) and (2.10) are multiplied by $(1-S)$.[47]

$$\begin{aligned}\mathbf{i}_{\mathrm{HOR}} = {} & (1-\mathrm{S})i_{0_{\mathrm{HOR}}}\left[\frac{p_{H_2}}{p_{H_2}^{\mathrm{ref}}}\exp\left(\frac{\alpha_a F}{RT}\left(\Phi_1 - \Phi_2 - U_0^{\mathrm{HOR}}\right)\right)\right. \\ & \left. - \exp\left(-\frac{\alpha_c F}{RT}\left(\Phi_1 - \Phi_2 - U_0^{\mathrm{HOR}}\right)\right)\right]\end{aligned} \tag{4.20}$$

$$\mathbf{i}_{\mathrm{ORR}} = -(1-\mathrm{S})i_{0_{\mathrm{ORR}}}\frac{p_{O_2}}{p_{O_2}^{\mathrm{ref}}}\exp\left(-\frac{\alpha_c F}{RT}\left(\Phi_1 - \Phi_2 - U_0^{\mathrm{ORR}}\right)\right) \tag{4.21}$$

Table 4-2: Liquid-Equilibrated Membrane Property Calculations 45

Parameter		Units	Equation
Membrane Liquid Water Volume Fraction	f_L		$f_L = \frac{\lambda_L \bar{V}_0}{\bar{V}_m + \lambda_L \bar{V}_0}$
Conductivity	κ_L	$\mathrm{S/cm}$	if $f_L \leq 0.45$ $\kappa_L = \frac{1}{2}(f_L - 0.06)^{1.5} \exp\left(\frac{15000}{R}\left(\frac{1}{T_{ref}} - \frac{1}{T}\right)\right)$ if $f_L > 0.45$ $\kappa_L = \frac{1}{2}(0.39)^{1.5} \exp\left(\frac{15000}{R}\left(\frac{1}{T_{ref}} - \frac{1}{T}\right)\right)$
Electroosmotic coefficient	ξ_L		$\xi_L = 2.55 \exp\left(\frac{400}{R}\left(\frac{1}{T_{ref}} - \frac{1}{T}\right)\right)$
Membrane transport coefficient	α_L	$\frac{\mathrm{mol}^2}{\mathrm{J \cdot cm \cdot s}}$	$\alpha_L = \frac{k_{\mathrm{sat}}}{\mu \bar{V}_w^2}\left(\frac{f}{f_L}\right)^2$
Hydrogen gas permeation coefficient	$\psi_{H_2,L}$	$\frac{\mathrm{mol}}{\mathrm{bar \cdot cm \cdot s}}$	$\psi_{H_2,L} = 1.8 \times 10^{-11} \exp\left(\frac{18000}{R}\left(\frac{1}{T_{ref}} - \frac{1}{T}\right)\right)$
Oxygen gas permeation coefficient	$\psi_{O_2,L}$	$\frac{\mathrm{mol}}{\mathrm{bar \cdot cm \cdot s}}$	$\psi_{O_2,L} = 1.2 \times 10^{-11} \exp\left(\frac{20000}{R}\left(\frac{1}{T_{ref}} - \frac{1}{T}\right)\right)$

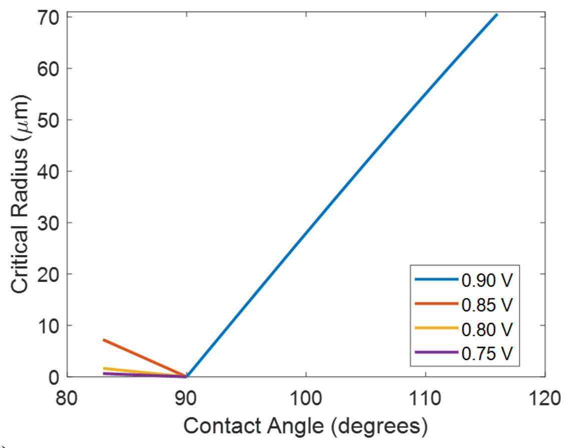
4.3 Multiphase Effects with Mechanical Degradation

A pinhole in the membrane allows for increased gas crossover which causes performance losses and leads to polymer degradation via radical attack. To determine if water condensation occurs in the pinhole, a critical radius can be calculated using Equation (4.4). The contact angle for Nafion can be hydrophilic or hydrophobic, depending on the water content of the membrane, the contact angle is 113 to 116° in dry conditions ($\lambda < 5$), less than 100° in saturated vapor, and 83 to 87° when in contact with liquid water.[51] When the surface of the pinhole is hydrophobic ($\theta_c > 90°$), the pinhole in the membrane will become filled with water when the pinhole radius is greater than the critical radius ($r_{hole} > r_c$). Likewise, when the surface of the pinhole is hydrophilic ($\theta_c < 90°$), the pinhole in the membrane will become filled with water when the pinhole radius is less than the critical radius ($r_{hole} < r_c$). A critical angle of 90° will result in a critical radius of 0 m, which represents the inflection point between the change between hydrophilic and hydrophobic behavior.

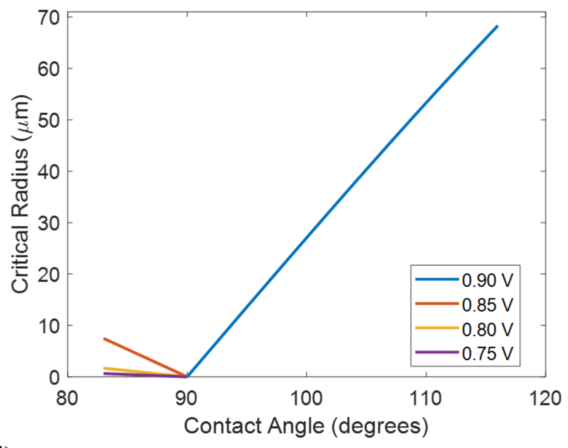
Figure 4-2 shows the results for the calculated critical radii at various voltage and RH values from multiphase simulations across the full range of critical angles reported for Nafion. The liquid pressure inside the pinhole was assumed to be the average of the pressure in the anode and cathode catalyst layers in order to calculate the capillary pressure, which is shown in Figure 4-3. The gas pressure inside the pinhole is assumed to be equal to the gas pressure in the catalyst layers, as the presence of the pinhole in the membrane will cause the pressures between the catalyst layers to equalize. The results in Figure 4-2 show the nonlinear relationship between voltage and RH with respect to the critical radius. The capillary pressure is positive at 0.9V and negative at lower potentials. For a positive capillary pressure, the critical radius can be calculated for hydrophobic contact angles, otherwise the pinhole will remain empty (hydrophilic pores are full and hydrophobic pores are empty for $p_c \leq 0$). Likewise, for negative capillary pressure, the critical radius can be calculated for hydrophilic contact angles.

For the range of operating conditions simulated, the critical radius for hydrophilic contact angles are all less than 10 μm. These values fall within the range of pinholes in which the gas crossover through the pinhole will be less than the gas crossover through permeation of the membrane (see Figure 3-1). For a hydrophilic membrane, pinholes smaller than the critical radius will fill up with liquid water, but larger pinholes will lead to a higher rate of gas crossover through the pinhole. Therefore, it is more favorable for the membrane to be slightly hydrophobic in order to cause the formation of liquid water in pinholes, which prevents the gas crossover rate from increasing due to mechanical degradation. In cases where the membrane is hydrophilic, which is typical at high water content values, the presence of liquid water in the membrane does not provide any mitigation of gas crossover; however, performance losses can still occur due to flooding.

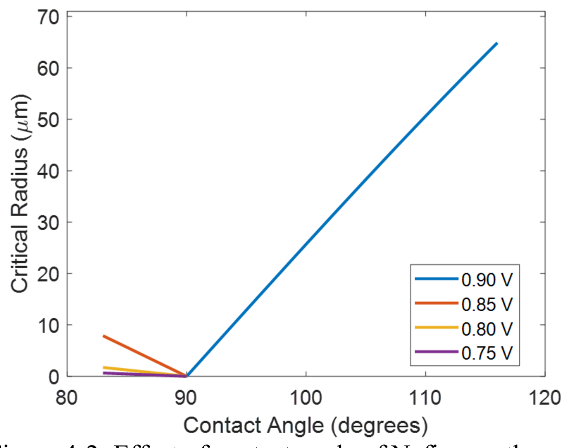
a)



b)



c)



d)

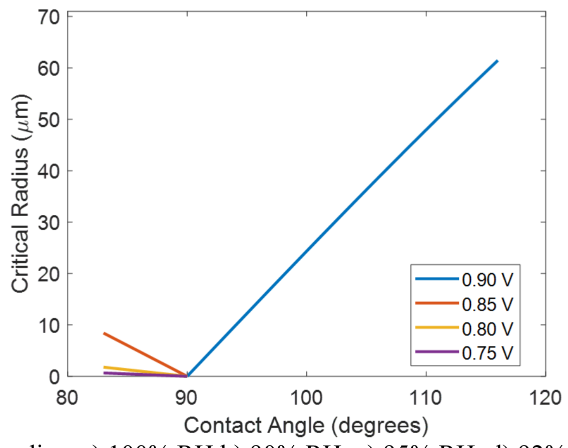


Figure 4-2: Effect of contact angle of Nafion on the critical radius, a) 100% RH b) 90% RH, c) 95% RH, d) 92% RH. Simulation conditions are 80^{o}C, 1 bar, feed/air stoichiometry 1.2/2.

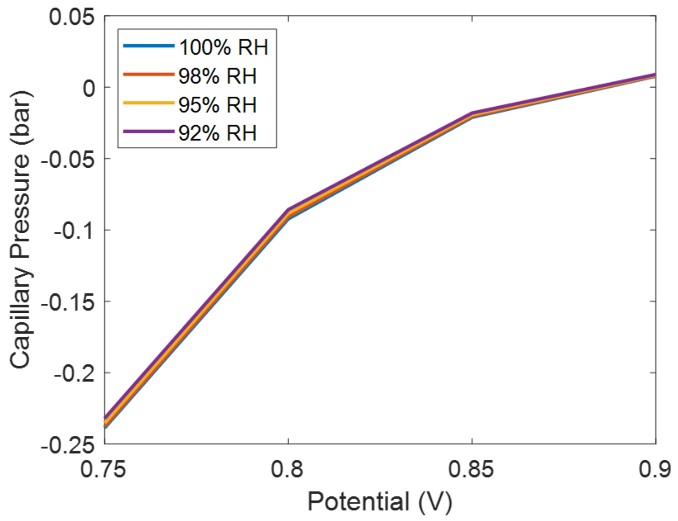
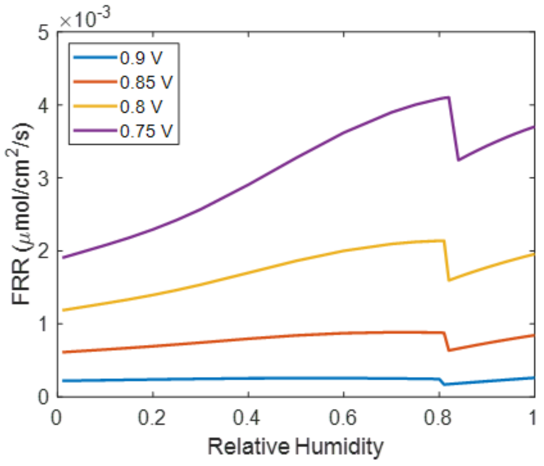


Figure 4-3: Capillary pressure in the membrane as a function of potential and RH. Simulation conditions are 80°C, 1 bar, feed/air stoichiometry 1.2/2.

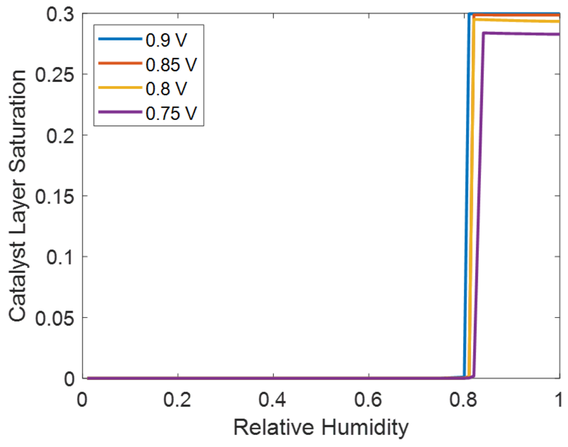
4.4 Multiphase Effects with Chemical Degradation

The empirical chemical degradation model is coupled with the multiphase fuel cell performance model to analyze the impact of flooding at high RH. The results in Figure 4-4a show that the FRR decreases with potential and increases with RH up until about 82% RH, at which point liquid water starts to form in the membrane. The formation of liquid water and decrease in FRR corresponds to a step change in the saturation of the catalyst layers as shown in Figure 4-4b, as the liquid water forms a film over the agglomerate and blocks access to reaction sites. This leads to sharp decrease in the FRR due to the lower catalyst surface area available for the two-electron oxygen reduction reaction which causes the formation of hydrogen peroxide and subsequently hydrogen fluoride. The loss of catalyst surface area is accounted for in Equations (4.20) and (4.21), by weighting the reaction rates by a factor of $(1 - S)$. After this drop, the FRR continues to increase with RH due to the increasing membrane conductivity at higher water content. For potentials below 0.9 V, the liquid-equilibrated FRR does not increase to above the FRR for the vapor-equilibrated membrane. The formation of liquid water also leads to a loss in performance, and Figure 4-4c shows that the change in the current density of the cell follows the same trends as the change in FRR. Therefore, at high RH conditions, some of the chemical degradation will be mitigated due to flooding, and the mitigation effects will increase at lower operating potentials. However, the degradation mitigation also occurs with a corresponding decrease in performance.

a)



b)



c)

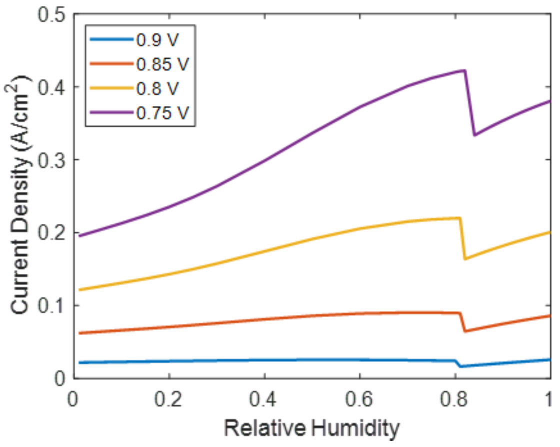


Figure 4-4: a) FRR as a function of RH and potential, b) mean saturation in the catalyst layers, c) current density as a function of RH and. Simulation conditions are 80°C, 1 bar, feed/air stoichiometry 1.2/2.

Chapter 5 – Cerium Mitigation of Membrane Chemical Degradation

Chemical degradation in PEMFC membranes is driven by the formation of hydroxide radicals which attack the polymer. Chemical scavengers, such as cerium and magnesium ions and their oxides, are added into the membrane to react with these hydroxide radicals and mitigate membrane degradation. Here a microkinetic model for chemical degradation of the polymer and mitigation via cerium ions in the membrane is incorporated into the fuel cell performance model. A model is developed for cerium transport throughout the cell, based on concentrated solution theory and calculated ion transport properties developed by Crothers *et al.*[73, 74]

5.1 Micro-Kinetic Degradation Model

A diagram of the modeling domain and location of key degradation species is shown in Figure 5-1. The modeling approach for the degradation of Nafion is based on the works of Wong and Kjeang and the reactions are listed in Table 5-1.[36, 37] Scheme 5-1 illustrates the different types of degradation of Nafion when attacked by hydroxyl radicals. All the reactions are assumed to be elementary steps, so that the reaction rates can be written as

$$r_h = k_h \sum_{i=1}^{n_h} c_i^{\nu_i} \tag{5.1}$$

where r_h, k_h, and n_h are the reaction rate, rate constant, and total number of reactants of reaction h, respectively, and c_i and ν_i are the concentration and stoichiometric coefficient of species i, respectively. The initial concentration of sulfonic-acid sites in the PEM is assumed to be equal to ρ_M/EW.

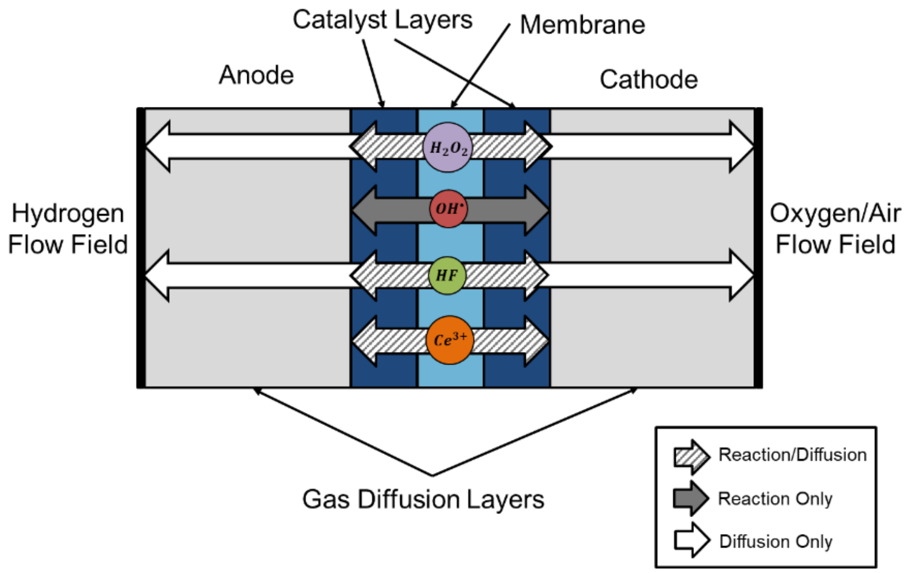
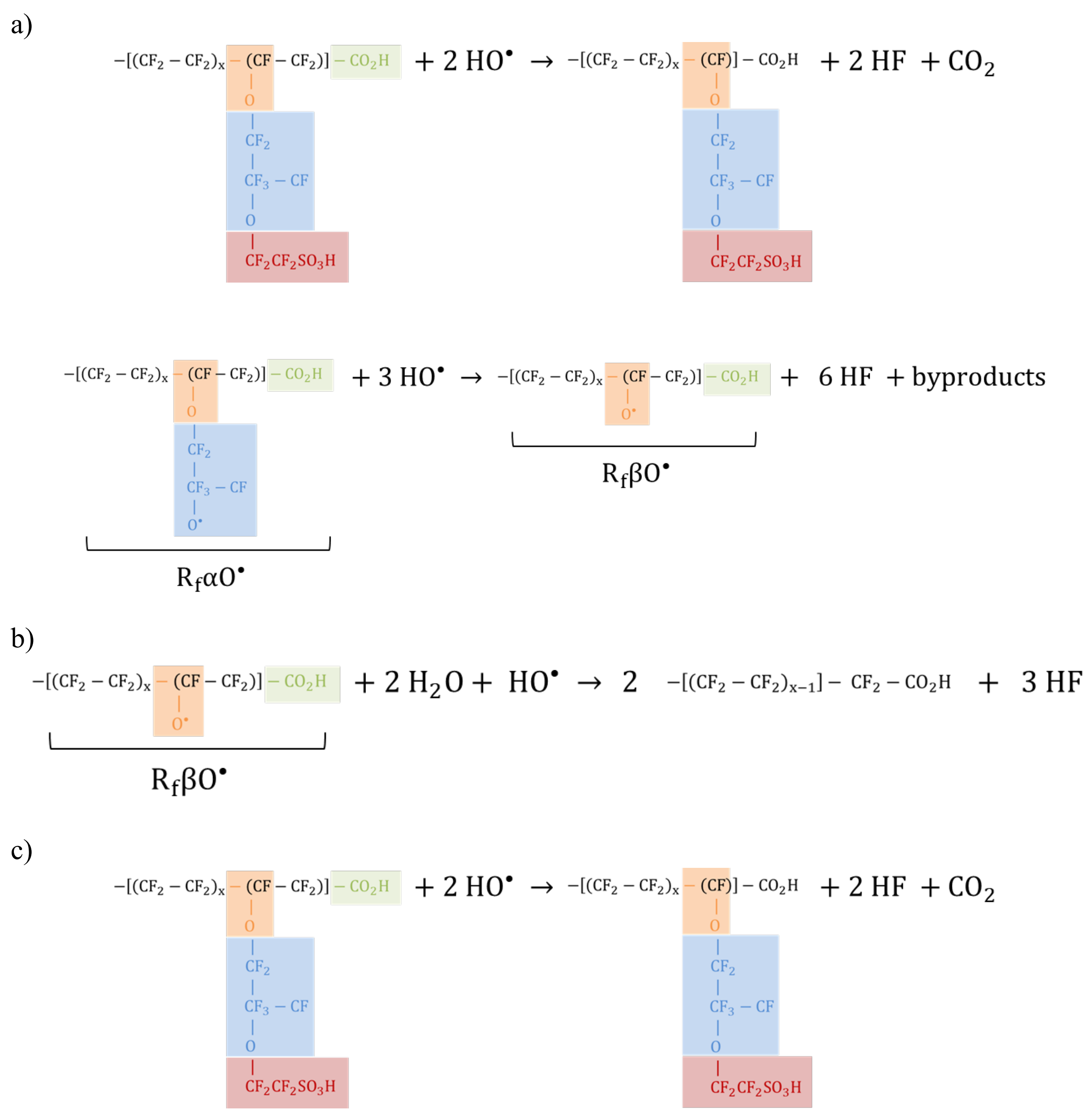


Figure 5-1: Diagram of the fuel-cell sandwich modeling domain and location of degradation species in the model.

The degradation process can be initialized by reaction of hydroxyl ions at the side-chain or the end-chain. Once the initial attack of hydroxyl ion on the side chain has occurred, leading to the degradation of the sulfonic-acid site, the remaining CF_2 groups in the side chain will also degrade until the main chain is reached and attacked. The kinetic equations are simplified in order to account for the total number of sulfonic-acid groups, end-chain groups, and fluoride ions that are present in the Nafion membrane and are released as a result of chemical degradation. The amount of fluoride ions that exits the PEMFC, called the fluoride release rate (FRR), is a measurement often used to quantify chemical degradation. In the model, the FRR is calculated as the sum of hydrogen-fluoride fluxes at the gas channels.



a)
−[(CF₂ − CF₂)ₓ − (CF − CF₂)] − CO₂H + 2 HO• → −[(CF₂ − CF₂)ₓ − (CF)] − CO₂H + 2 HF + CO₂
O, CF₂, CF₃ − CF, O, CF₂CF₂SO₃H
+ 3 HO• → + 6 HF + byproducts
R_fαO•
R_fβO•
b)
−[(CF₂ − CF₂)ₓ − (CF − CF₂)] − CO₂H + 2 H₂O + HO• → 2 −[(CF₂ − CF₂)ₓ₋₁] − CF₂ − CO₂H + 3 HF
R_fβO•
c)
−[(CF₂ − CF₂)ₓ − (CF − CF₂)] − CO₂H + 2 HO• → −[(CF₂ − CF₂)ₓ − (CF)] − CO₂H + 2 HF + CO₂

Scheme 5-1: Degradation mechanisms: a) side-chain degradation/reactions 2 & 3, b) chain scission/reaction 4, c) end-chain degradation/unzipping/reaction 5.

Table 5-1: Membrane Degradation Reaction Kinetics

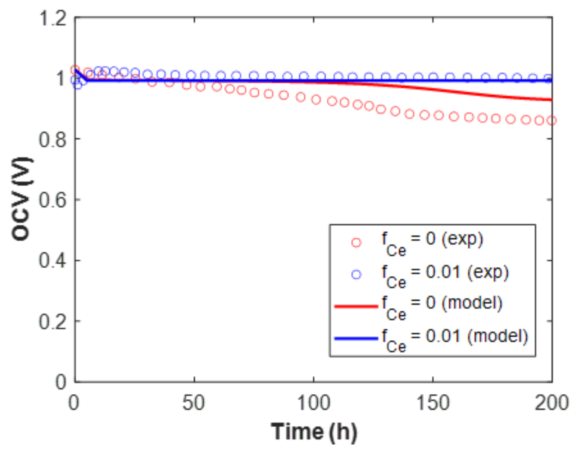
Reaction Number		**Rate constant**	**Ref.**
1	$H_2O_2 \rightarrow 2HO^\bullet$	$3 \times 10^{-3} s^{-1}$	fit
2	$R_fSO_3 + HO^\bullet \rightarrow R_f\alpha O^\bullet + 4HF$	$3.7 \times 10^{6}\ M^{-1}s^{-1}$	[36, 37]
3	$R_f\alpha O^\bullet + 3HO^\bullet \rightarrow R_f\beta O^\bullet + 6HF$	$3.75 \times 10^{7}\ M^{-1}s^{-1}$	[36, 37]
4	$R_f\beta O^\bullet + 2H_2O + HO^\bullet \rightarrow 2R_fCOOH + 3HF$	$7.5 \times 10^{7}\ M^{-1}s^{-1}$	[37]
5	$R_fCOOH + 2HO^\bullet \rightarrow R_fCF_2 + 2HF$	$5.8 \times 10^{6}\ M^{-1}s^{-1}$	[36, 37]
6	$Ce^{3+} + HO^\bullet + H^+ \rightarrow Ce^{4+} + H_2O_2$	$1 \times 10^{11}\ M^{-1}s^{-1}$	fit

The gas crossover rate through the membrane drives chemical degradation, as hydrogen and oxygen react to form hydrogen peroxide and hydroxyl radicals. The gas crossover rate rates are calculated in Equation (2.17) and the initial values for the membrane permeation coefficients for hydrogen and oxygen are listed in Table 2-4. To account for the impact of membrane degradation on gas crossover, a polynomial function was fit to the data of Coms *et al.*[22] The modified permeation coefficients are,

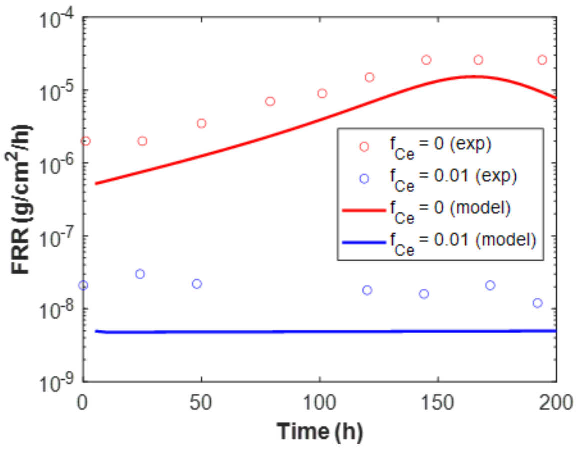
$$\frac{\psi_i}{\psi_{i,0}} = 102\left(\frac{c_{R_fSO_3}}{\frac{\rho_M}{EW}}\right)^2 - 201\left(\frac{c_{R_fSO_3}}{\frac{\rho_M}{EW}}\right) + 100 \tag{5.2}$$

where $c_{R_fSO_3}$ the concentration of sulfonic-acid sites in the membrane. The experimental data used for fitting is shown in Figure 5-2. The increase in gas crossover can be attributed to changes in the morphology of the ionomer as it degrades, including an increase in the size of hydrophilic domains and formation of microvoids where localized degradation has caused a loss of ionomer.[75, 76]

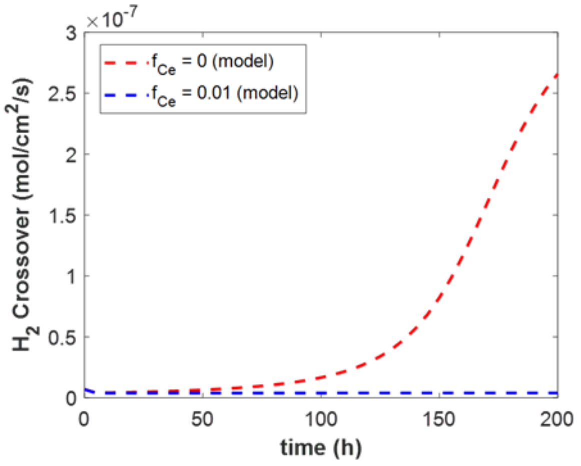
a)



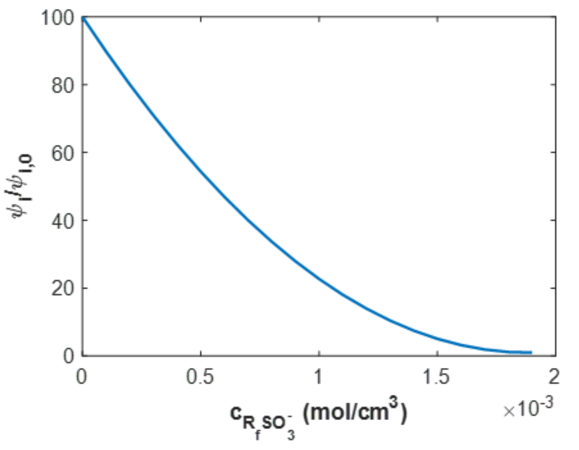
b)



c)



d)



e)

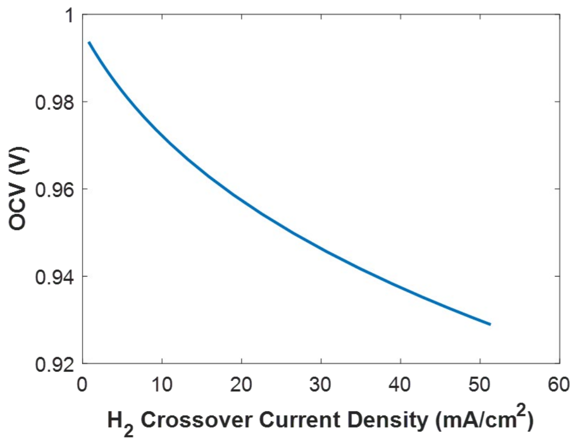


Figure 5-2: A comparison of simulation and experimental measurements for a) OCV and b) FRR and corresponding c) predicted hydrogen crossover rates. Solid lines are simulation results and open circles are experimental results from Coms, *et al*.[22] d) Resulting fit for permeation coefficients as a function of sulfonic acid group concentration. e) Relationship between OCV and hydrogen crossover current density. Simulation conditions are 95°C, 1.5 bar, 50%/50% RH at anode/cathode, stoichiometry 5.0/5.0 air/feed based on a current density of 0.2 A/cm^2.

The degradation products H_2O_2 and HF are generated in the catalyst layers and membrane and are allowed to diffuse into to the GDLs and out of the PEMFC. The rate of diffusion for these species is calculated using Fick's law and the diffusion coefficients are listed in **Table 5-2:** Diffusion Coefficients of H2O2 and HFTable 5-2. Due to the short lifetime of the radical species, the diffusion distance can be approximated as zero.[26, 34]

Table 5-2: Diffusion Coefficients of H_2O_2 and HF

	Diffusion Coefficient (cm²/s)	**Ref**
$\mathcal{D}_{\mathrm{H_2O_2,M}}$	1.5×10^{-6}	34
$\mathcal{D}_{\mathrm{H_2O_2,GDL}}$	0.188	36
$\mathcal{D}_{\mathrm{HF,M}}$	1.5×10^{-6}	34
$\mathcal{D}_{\mathrm{HF,GDL}}$	0.26	36

5.2 Modeling of Cerium-Doped Membranes

In order to mitigate chemical degradation of the ionomer, cerium ions react with hydroxyl radicals according to reaction 6 in Table 5-1. While cerium may react with other degradation products, the reaction of Ce^{3+} with hydroxyl ions is the primary reaction pathway for cerium.[35] In this model, the cerium in the membrane is assumed to be present only in the 3+ charge state and the concentration of cerium ions in the 4+ charge state is considered to be equal to approximately zero. The operating potential for the PEMFC results in a high overpotential for the Ce^{3+}/Ce^{4+} redox reaction ($U^0 = 1.44$ V vs. SHE), driving cerium ions into the 3+ charge state.[35, 38] A more complete analysis of the Ce^{3+}/Ce^{4+} redox couple in PFSA membranes is analyzed by Gubler and Kopponel, who demonstrated that >99.99% of cerium ions are present in the 3+ charge state.[35] The model can be extended to include the effects of Ce^{4+}, which may be an important consideration for analysis of start/stop cycles, by modifying the concentrated-solution-theory equations. However, since the contributions of these reactions to the overall mitigation are small compared to the reaction of Ce^{3+}, they are not considered in this model.

The dependence of water uptake on the concentration of cerium is calculated using a polynomial fit of cerium content and water activity from Baker *et al.,*[77]

$$\lambda = 1.426 + 9.88a + 0.1256f_{Ce} - 14.73a^2 + 2.826af_{Ce} + 14.42a^3 - 4.0406a^2f_{Ce} \quad (5.3)$$

where λ is the water content, a is the water activity, and f_{Ce} is the fraction of sulfonic acid sites in the membrane that are occupied by cerium ions,

$$f_{\mathrm{Ce}} = \frac{z_{Ce^{3+}}c_{\mathrm{Ce}^{3+}}}{\rho_M/EW} \quad (5.4)$$

where ρ_M is the dry membrane density and EW is the equivalent weight of the polymer (1100 g/mol). Using this definition, a membrane that is fully saturated with protons would have a cerium content of $f_{Ce} = 0$ and a membrane that is fully saturated with cerium ions would have a cerium content of $f_{Ce} = 1$. Water uptake curves as a function of cerium content are shown in Figure 5-3. An illustration of cerium in the membrane and how f_{Ce} is calculated is shown in Figure 5-4.

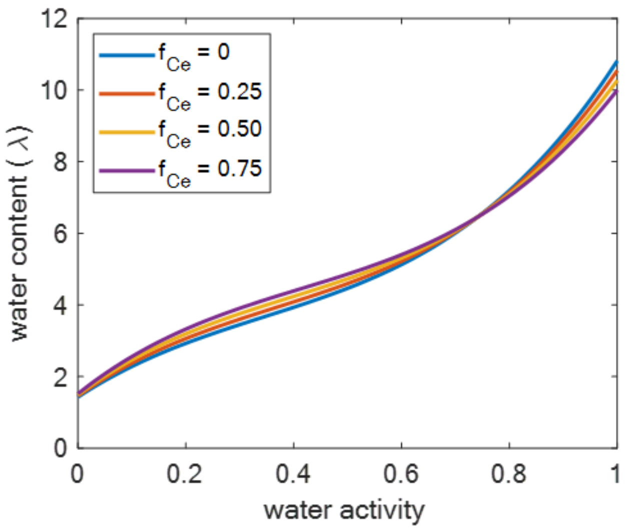


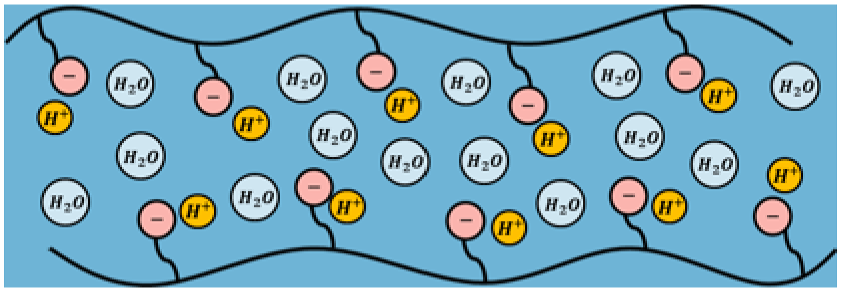
Figure 5-3: Water uptake dependence on cerium content.

Assuming that the cerium cannot leave the ionomer, an additional mass balance is required to determine the concentration profile within the membrane phase of the PEMFC,

$$\int_0^x c_{Ce} dx = n_{Ce} \tag{5.5}$$

where n_{Ce} is the total number of moles of cerium ions initially present in the membrane at the beginning-of-life and x is the distance across the membrane and catalyst layers. This formulation ensures conservation of the mass of cerium inside of the membrane-electrode assembly (MEA). Experiments have shown the presence on cerium in the PEMFC effluent, indicating that cerium can leave the cell via ion pairing.[78] However, the amount of cerium that exits the cell is very small, on the order of ng/cm^2 over the course of 1000 hours (compared to a typical loading on the order of μg/cm^2),[78] and is therefore neglected in the model.

a)



b)

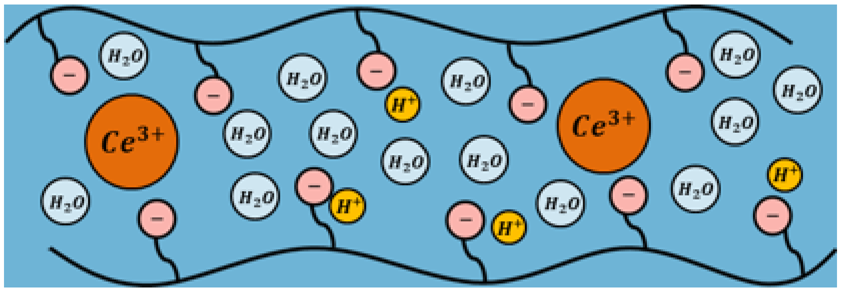


Figure 5-4: Concentrated solution theory model with (a) membrane, water and protons (f_{Ce}), (b) membrane doped with cerium ions. In the representative volume of (b) with 10 SO_3^- groups, 2 Ce^{3+} ions complex with 6 SO_3^- groups, resulting a f_{Ce} value of 0.6.

5.3 Cerium-Ion Transport Model

The equations describing transport of water and protons through the membrane are derived from concentrated-solution theory, where the membrane acts as the reference velocity (i.e. zero velocity relative to the laboratory frame of reference for negligible swelling rate).[42, 43] For a multicomponent system that is isothermal and isotropic, transport of all mobile species i obey

$$c_i \nabla \mu_i = K_{i\mathrm{M}}(-\mathbf{v}_i) + \sum_{j \neq i,M} K_{ij}(\mathbf{v}_j - \mathbf{v}_i) \tag{5.6}$$

where c_i, μ_i, and $\mathbf{v}_i$ are the concentration, chemical potential, and velocity of species i, respectively, K_{ij} is the friction coefficient between species i and j, and K_{iM} is the friction coefficient between species i and the membrane.[68] To satisfy the Gibbs-Duhem equation, for the membrane,

$$c_\mathrm{M} \nabla \mu_\mathrm{M} - \nabla p = \sum_{i \neq \mathrm{M}} K_{i\mathrm{M}} \boldsymbol{v}_i \tag{5.7}$$

where p is the pressure. This results in $N-1$ independent equations for a system with N species. Onsager's reciprocal relations show that $K_{ij} = K_{ji}$, therefore a system with N species will have $N(N-1)/2$ friction coefficients.[79] The friction coefficients can be related to the binary diffusion coefficients by

$$\mathcal{D}_{ij} = \frac{RTc_ic_j}{K_{ij}c_T} \tag{5.8}$$

where $\mathcal{D}_{ij}$ is the binary diffusion coefficient of species i and j, R is the ideal gas constant, T is the temperature, and c_T is the total molar concentration of the solution.[68]

For the system of interest, which is 1D across the PEMFC sandwich, Equation (5.6) in matrix form is

$$\boldsymbol{D} = \boldsymbol{M}^{\mathrm{M}}\boldsymbol{V} \tag{5.9}$$

where $\boldsymbol{D}$ is the vector of driving forces with length $N-1$, $\boldsymbol{V}$ is the vector of velocities with length $N-1$, and $\boldsymbol{M}^{\mathrm{M}}$ is the transport coefficient matrix with dimensions $N-1$ by $N-1$. Equation (5.7) is excluded from the matrix (i.e. it is the Nth eqution) because it depends on the $N-1$ instances of Equation (5.6) The superscript M denotes that the reference velocity is that of the membrane. The entries of the matrix are $D_i = c_i\nabla\mu_i$, $V_i = \mathbf{v}_i$, and $M_{ij}^{M} = K_{ij}$ for $i \neq j$ and $M_{ii}^{M} = -\sum_{j\neq i} K_{ij}$.

In an isothermal system, inverting Equation (5.9) relates the flux of species i to a linear combination of non-membrane electrochemical potentials,

$$\mathbf{N}_i = -\sum_{j\neq \mathrm{M}} L_{ij}^{\mathrm{M}} c_i c_j \nabla\mu_j \tag{5.10}$$

where c_i is the concentration of species i, $\mathbf{N}_i$ is the molar flux vector of species i, and L_{ij}^{M} is the transport coefficient for species i and j.[42] The matrix $\boldsymbol{L}^{M}$ with entries L_{ij}^{M} is symmetric and has dimensions $N-1$ by $N-1$, where N is the total number of species in the system (including the membrane). $\boldsymbol{L}^{M}$ is defined as

$$\boldsymbol{L}^{M} = -(\boldsymbol{M}^{M})^{-1} \tag{5.11}$$

The L_{ij}^{M} transport coefficients are not measured directly because experimental conditions that isolate each of these coefficients are not practical. To use this system of equations, the transport coefficient matrix must be rewritten in terms of measurable properties.[73, 80]

$$\mathbf{N}_i = -\sum_{j\neq \mathrm{M}} \left(\alpha_{ij}^{\mathrm{M}} + \frac{t_i^{\mathrm{M}} t_j^{\mathrm{M}} \kappa}{z_i z_j F^2}\right)\nabla\mu_{j,n} - \frac{t_i^{\mathrm{M}}\kappa}{z_i F^2}\frac{\nabla\mu_n}{z_n} \tag{5.12}$$

where F is Faraday's constant, z_i is the valance of species i, and $\mu_{i,n} = \mu_i - \frac{z_i}{z_n}\mu_n$ is the chemical of species i relative to charged species n, α_{ij}^{M} is the transport coefficient for species i and j, t_i^{M} is the transference number for species i, κ is the ionic conductivity, and ξ is the electroosmotic coefficient.[81] α_{ij} is symmetric ($\alpha_{ij} = \alpha_{ji}$) and is similar to a generalized effective diffusion coefficient and describes the flux of i due to a chemical potential gradient of $\mu_{i,n}$ in the absence of current. $\mu_{i,n}$ quantifies the chemical potential of species i and, since it is taken relative to charged species n, is independent of electric potential. This prevents the use of an arbitrary definition of $\nabla\Phi$ when there are concentration gradients. μ_n is the only term that depends on the electric potential in the membrane. The relationship between these properties and the entries in $\boldsymbol{L}^M$ are

$$\kappa = F^2 \sum_{i \neq \mathrm{M}} \sum_{j \neq \mathrm{M}} L_{ij}^{\mathrm{M}} z_i c_i z_j c_j \tag{5.13}$$

$$t_i^M = \frac{z_i c_i F^2}{\kappa} \sum_{j \neq \mathrm{M}} L_{ij}^{\mathrm{M}} z_j c_j \tag{5.14}$$

$$\xi = \frac{t_w^{\mathrm{M}}}{z_w} \tag{5.15}$$

$$\alpha_{ij}^{\mathrm{M}} = L_{ij}^{\mathrm{M}} c_i c_j - \frac{t_i^{\mathrm{M}} t_j^{\mathrm{M}} \kappa}{z_i z_j F^2} \tag{5.16}$$

Note that the electro-osmotic coefficient has a finite value although $z_w = 0$ (upon substitution of Equation (5.14) for $i = w$ into Equation (5.15), the z_w term cancels out).[81]

Reactions at the electrodes in the PEMFC involve protons and, as such, the electric potential is typically quantified by the electrochemical potential of a proton (i.e. a proton reference electrode). Therefore, a convenient choice for the electrochemical reference species n is the proton, so that $\mu_n = \mu_{\mathrm{H}} = F\Phi$. Using this definition, Equation (5.12) with protons (H) set as the reference species, yields for protons,

$$\mathbf{N}_{\mathrm{H}} = -\left(\alpha_{\mathrm{HCe}}^{\mathrm{M}} + \frac{t_{\mathrm{H}}^{\mathrm{M}} t_{\mathrm{Ce}}^{\mathrm{M}} \kappa}{z_{\mathrm{H}} z_{\mathrm{Ce}} F^2}\right) \nabla\mu_{\mathrm{Ce,H}} - \left(\alpha_{\mathrm{H}w}^{\mathrm{M}} + \xi \frac{t_{\mathrm{H}}^{\mathrm{M}} \kappa}{z_{\mathrm{H}} F^2}\right) \nabla\mu_w - \frac{t_{\mathrm{H}}^{\mathrm{M}} \kappa}{z_{\mathrm{H}} F} \nabla\Phi \tag{5.17}$$

for cerium,

$$\mathbf{N}_{\mathrm{Ce}} = -\left(\alpha_{\mathrm{CeCe}}^{\mathrm{M}} + \left(\frac{t_{\mathrm{Ce}}^{\mathrm{M}}}{z_{\mathrm{Ce}}}\right)^2 \frac{\kappa}{F^2}\right) \nabla\mu_{\mathrm{Ce,H}} - \left(\alpha_{\mathrm{Ce}w}^{\mathrm{M}} + \xi \frac{t_{\mathrm{Ce}}^{\mathrm{M}} \kappa}{z_{\mathrm{Ce}} F^2}\right) \nabla\mu_w - \frac{t_{\mathrm{Ce}}^{\mathrm{M}} \kappa}{z_{\mathrm{Ce}} F} \nabla\Phi \tag{5.18}$$

and for water,

$$\mathbf{N}_w = -\left(\alpha_{w\mathrm{Ce}}^{\mathrm{M}} + \xi \frac{t_{\mathrm{Ce}}^{\mathrm{M}} \kappa}{z_{\mathrm{Ce}} F^2}\right) \nabla\mu_{\mathrm{Ce,H}} - \left(\alpha_{ww}^{\mathrm{M}} + \xi^2 \frac{\kappa}{F^2}\right) \nabla\mu_w - \xi \frac{\kappa}{F} \nabla\Phi \tag{5.19}$$

To relate $\mu_{\mathrm{Ce,H}}$ to the species concentrations, an ideal solution for cerium, protons and the membrane is assumed,

$$\nabla \mu_{\mathrm{Ce,H}} = \frac{RT}{c_{\mathrm{Ce}}} \nabla c_{\mathrm{Ce}} - \left(\frac{z_{\mathrm{Ce}}}{z_{\mathrm{H}}}\right) \frac{RT}{c_{\mathrm{H}}} \nabla c_{\mathrm{H}} \tag{5.20}$$

The friction factors are calculated using the theory of multi-ion transport developed by Crothers *et al.*,[73, 74] which is summarized in Section 5.4.

5.4 Calculation of Friction Factors

The PEM is considered a mixture of protons, cerium ions, water, and membrane charged sites in the hydrophilic, water-filled domains of the membrane. The various types of interactions in this system includes ion/solvent, ion/ion, ion/membrane, and solvent/membrane, where water is taken to be the solvent. A hydrodynamic model of the membrane pores is used to calculate the ion/membrane and solvent/membrane friction coefficients.[73] These friction coefficients are scaled by the volume fraction of polymer ε_M and the tortuosity τ_M, which is calculated as

$$\tau_M = \left(1 - \varepsilon_M - \varepsilon^{crit}\right)^{-0.95} \tag{5.21}$$

where in the membrane and ε^{crit} is the critical volume fraction, the point below which the water content is too low for transport to effectively take place,[77]

$$\varepsilon^{crit} = 0.47 f_{\mathrm{Ce}} + 0.082 - \frac{(0.43 f_{Ce} - 0.016)}{1 + \exp(-100 f_{Ce} + 1.84)} \tag{5.22}$$

For the ion/solvent friction coefficients, the Stokes-Einstein equation is used to take into account the drag of an ion with the water in the hydrophilic domain of the membrane,

$$\mathcal{D}_{iw} = \frac{(1 - \varepsilon_M)}{\tau_M} \frac{\eta^{\infty}}{\eta} \mathcal{D}_{iw}^{\infty} \tag{5.23}$$

where η is the viscosity and ∞ denotes infinite dilution. Einstein's velocity equation corrects for changes in viscosity with concentration,

$$\eta = \frac{\eta^{\infty}\left(1 + \sum_{i \neq w,M}^{N} \frac{c_i \tilde{V}}{2}\right)}{\left(1 - \sum_{i \neq w,M}^{N} c_i \tilde{V}_i\right)^2} \tag{5.24}$$

where $\tilde{V}_i$ is the effective molar viscous volume of species i and is specified by bulk-electrolyte measurements.

For ion/ion interactions, Debye-Hückel-Onsager theory predicts that in binary electrolytes the diffusion coefficient varies with the square-root of the concentration for oppositely charged ions,

$$\mathcal{D}_{ij} = D_{iw} \sqrt{I} \quad \text{for} \quad z_{i \neq M,w} z_{j \neq M,w} < 0 \tag{5.25}$$

where I is the ionic strength,

$$I = \frac{1}{2} \sum_{i \neq 0,M}^{N} c_i \, z_i^2 \tag{5.26}$$

For similarly charged ions, the diffusion coefficient approaches infinity.

$$\mathcal{D}_{ij} \rightarrow \infty \quad \text{for} \quad z_{i \neq M,w} z_{j \neq M,w} > 0 \tag{5.27}$$

To determine the ion/membrane interactions, an expression is derived to satisfied both a microscale hydrodynamic model (e.g. Navier-Stokes) as well as the Stefan-Maxwell-Onsager relations,

$$K_{iM} = w_i \mathcal{K}_i + \sum_{j \neq M}^{N} K_{ij} \left(\frac{\mathcal{K}_i}{\mathcal{K}_j} - 1 \right) \tag{5.28}$$

where w_i and $\mathcal{K}_i$ are the mass fraction and hydrodynamic friction coefficient of species i, respectively. The hydrodynamic friction coefficient is calculated as

$$\mathcal{K}_i = \frac{4G\eta}{R_{\text{pore}}^2 \theta_i} \left(\frac{\tau_M}{1 - \varepsilon_M} \right) \tag{5.29}$$

where G is the geometric factor, θ_i is a function describing how species i distributes across the channel, and R_{pore} is the pore radius and is a function of the polymer volume fraction,

$$R_{\text{pore}} = R_{pore,0} \frac{1}{2} \varepsilon_M^{-m} (1 - \varepsilon_M)^{\frac{1}{2}} \tag{5.30}$$

where $R_{\text{pore,0}}$ is the dry domain spacing and m is a swelling parameter. The function θ_i is calculated by solving the linearized Poisson-Boltzmann equation for the distribution of ionic species across a cylindrical membrane pore with radius $R_{\text{pore}} - 2R_w$, where $2R_w = 0.275$ nm is the diameter of a water molecule. The pore is assumed to have sulfonate groups evenly distributed along the channel walls.

$$\theta_{i \neq \text{w}} = \beta^2 \left(2 - \beta^2 - z_i \varrho \left(\beta^2 + \frac{8\varrho}{\left(R_{\text{pore}} k\right)^2} - \frac{4\beta I_0\left(R_{\text{pore}} k\right)}{R_{\text{pore}} k I_1\left(R_{\text{pore}} k\right)} \right) \right) \tag{5.31}$$

where $\beta = \frac{(R_{\text{pore}} - 2R_0)}{R_{\text{pore}}}$, $\varrho = \frac{\sum_{i \neq M}^{N} n_i z_i}{\sum_{i \neq M}^{N} n_i z_i^2}$, I_0 and I_1 are modified Bessel functions of the first kind with order 0 and 1, respectively, and k is the inverse Debye length,

$$k = \left(\sum_{i \neq M}^{N} \frac{c_i z_i^2 F^2}{\varepsilon_r \varepsilon_0 RT} \right)^{\frac{1}{2}} \tag{5.32}$$

where ε_r is the bulk solvent dielectric constant and ε_0 is vacuum permittivity.

The parameters used to calculate the friction factors in the model are listed in Table 5-3. Figure 5-5 shows the calculated transport coefficients for a range of water content and cerium concentrations.[77]

Table 5-3: Parameters for Calculation of Friction Factor Coefficients

Property		Units	Value	Ref
Diffusivity of cerium in water	$\mathcal{D}_{Ce,w}$	m^2/s	6.2×10^{-10}	[82]
Diffusivity of protons in water	$\mathcal{D}_{H,w}$	m^2/s	9.31×10^{-9}	[82]
Effective molar viscous volume of cerium	$\tilde{V}_{Ce}$	m^3/mol	0.1543	[83]
Effective molar viscous volume of hydrogen	$\tilde{V}_H$	m^3/mol	0.0213	[84]
Dry domain spacing	$R_{\text{pore},0}$	nm	2.7	[51]
Swelling parameter	m		1.33	[51]
Geometric factor	G		4	[73]
Bulk solvent dielectric constant	ε_r	F/m	78.3	[73]

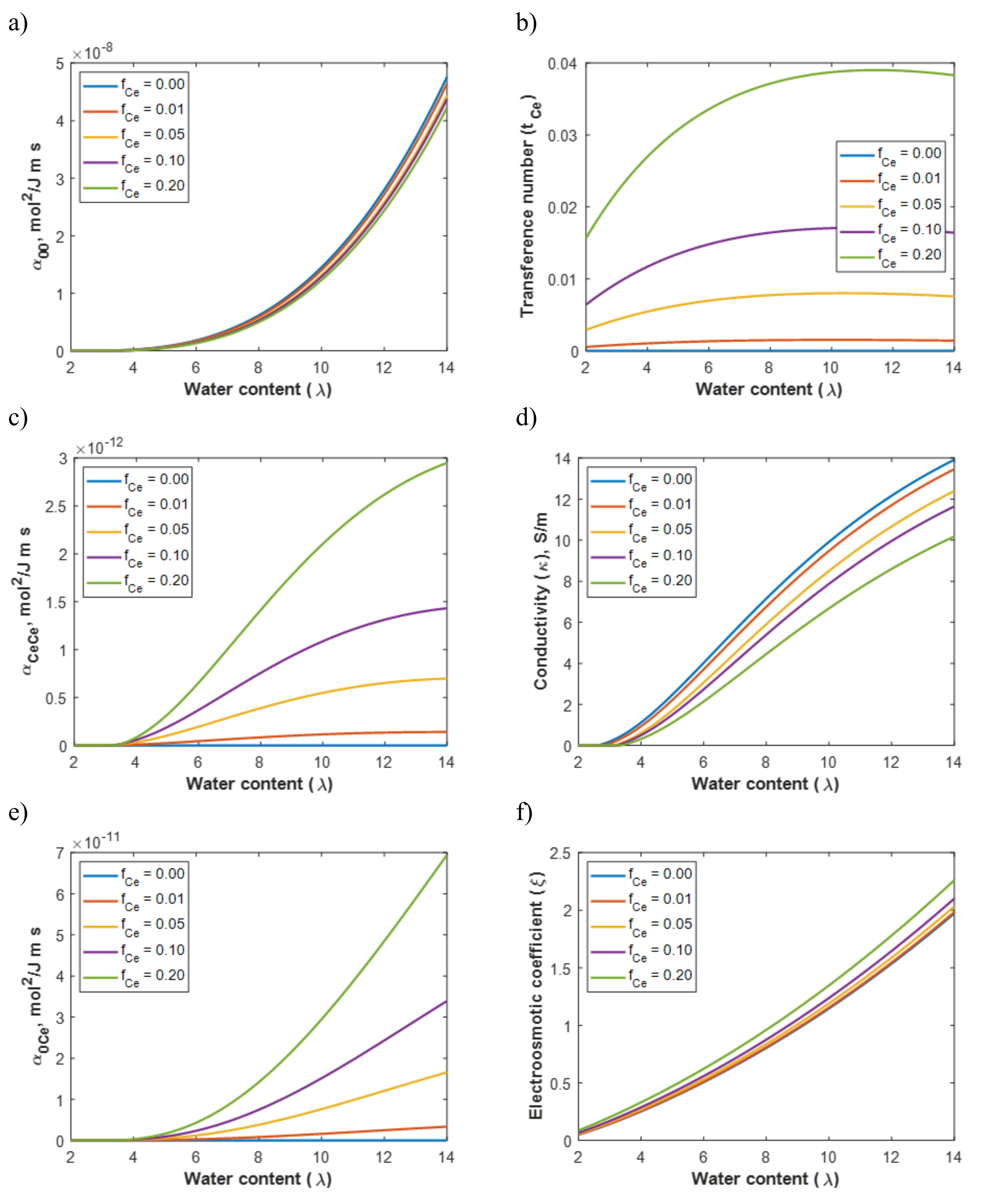


Figure 5-5: Membrane properties as a function of water and cerium content, (a) water-water transport coefficient, (b) transference number, (c) cerium-cerium transport coefficient, (d) conductivity, (e) water-cerium transport coefficient, and (f) electroosmotic coefficient, as calculated by concentrated solution theory.

5.5 Cerium-ion Impact on Reaction Kinetics

HOR and ORR that predominantly occur at the anode and cathode, respectively, can in reality occur at either electrode due to the crossover of $\mathrm{H_2}$ and $\mathrm{O_2}$ through the membrane. In addition, the two-electron ORR can take place and form hydrogen peroxide. Butler-Volmer kinetics are used for HOR and Tafel kinetics are used for ORRs,

$$\mathbf{i}_{\mathrm{HOR}} = \mathrm{i}_{0_{\mathrm{HOR}}}\left[\frac{p_{\mathrm{H_2}}}{p_{\mathrm{H_2}}^{\mathrm{ref}}}\exp\left(\frac{\alpha_a F}{RT}\left(\Phi_1 - \Phi_2 - \mathrm{U}_0^{\mathrm{HOR}}\right)\right) - \left(\frac{a_{HM}}{a_{HM}^{ref}}\right)^2 \exp\left(-\frac{\alpha_c F}{RT}\left(\Phi_1 - \Phi_2 - \mathrm{U}_0^{\mathrm{HOR}}\right)\right)\right] \tag{5.33}$$

$$\mathbf{i}_{\mathrm{ORR_{4e^-}}} = -\mathrm{i}_{0_{\mathrm{ORR_{4e^-}}}}\frac{p_{\mathrm{O_2}}}{p_{\mathrm{O_2}}^{\mathrm{ref}}}\left(\frac{a_{HM}}{a_{HM}^{ref}}\right)^4 \exp\left(-\frac{\alpha_c F}{RT}\left(\Phi_1 - \Phi_2 - \mathrm{U}_0^{\mathrm{ORR_{4e^-}}}\right)\right) \tag{5.34}$$

$$\mathbf{i}_{\mathrm{ORR_{2e^-}}} = -\mathrm{i}_{0_{\mathrm{ORR_{2e^-}}}}\frac{p_{\mathrm{O_2}}}{p_{\mathrm{O_2}}^{\mathrm{ref}}}\left(\frac{a_{HM}}{a_{HM}^{ref}}\right)^2 \exp\left(-\frac{\alpha_c F}{RT}\left(\Phi_1 - \Phi_2 - \mathrm{U}_0^{\mathrm{ORR_{2e^-}}}\right)\right) \tag{5.35}$$

where $\mathrm{i}_{0,\mathrm{HOR}}$, $\mathrm{i}_{0,\mathrm{ORR_{4e^-}}}$, and $\mathrm{i}_{0,\mathrm{ORR_{2e^-}}}$ are the respective exchange current densities, α_a and α_c are the anode and cathode coefficients, $\mathrm{U}_0^{\mathrm{HOR}}$, $\mathrm{U}_0^{\mathrm{ORR_{4e^-}}}$, and $\mathrm{U}_0^{\mathrm{ORR_{2e^-}}}$ are the respective standard potentials, p_i and p_i^{ref} are the partial pressure and reference pressure of species i, and a_{HM} and a_{HM}^{ref} are the proton activity and reference proton activity, respectively, R is the ideal gas constant, and T is the absolute temperature.[28] The proton activity is taken to be the fraction of membrane sulfonic-acid sites that are occupied by protons. The reference value for proton activity is that of protons in unexchanged Nafion and is taken to be equal to 1. In previous chapters the proton activity is assumed to be equal to the reference state, so the ratio a_{HM}/a_{HM}^{ref} is equal to 1. In the case where cerium is present in the membrane, the ratio a_{HM}/a_{HM}^{ref} reduces to the mole fraction of protons occupying sulfonic acid sites, which is equivalent to $1 - f_{Ce}$.

5.6 Model Solution

The model is run in MATLAB (see Appendix B for codes used). To initialize the simulation, certain operating parameters such as temperature, pressure, feed stoichiometry, air stoichiometry, membrane properties, initial cerium doping, *etc.* must be specified. These parameters are used to calculate the initial condition for the transient simulation by solving the PEMFC model under steady-state conditions. Furthermore, the cerium is assumed to be present in the membrane only at uniform concentration, and the initial cerium flux is zero. The governing equations are constructed using a finite-volume method approach, which enforces conservation of mass and energy. The system of equations is solved using a multidimensional Newton-Raphson technique (Band(J)) developed by Newman,[52, 68] which is detailed in Appendix C of Newman and Thomas-Alyea.[68] Each domain in the model is discretized using 40 mesh points. The full list of equations and boundary conditions are listed in Appendix A. The simulation must be initialized by specifying certain conditions, including: cell current or cell voltage, RH in the hydrogen and air feeds, stoichiometry of feed or feed rates, temperature, pressure in the gas channels, membrane thickness, and cerium content (f_{Ce}), To incorporate transient effects, a Crank-Nicolson approach is used to calculate the time derivatives in the mass- and energy-balance equations.

To explore the impact of the various contribution to the overpotentials, a voltage-loss breakdown was calculated by removing limiting factors to PEMFC performance sequentially from the final polarization curve. The transport losses attributed to cerium are divided into two categories. The first is effects that the cerium ions have in changing the transport properties of water and protons. The second is losses that occur due to the reduction in proton activity in the membrane phase, which is included in the kinetics in Equations (5.33)-(5.35). To remove this limitation, we set the ratio $a_{\text{HM}}/a_{\text{HM}}^{\text{ref}} = 1$, which assumes a membrane with zero cerium content. Mass-transport limitations occur when the PEMFC starts to become reactant limited. To remove mass-transport limitations, the simulation is run at a high stoichiometry for hydrogen gas and air. The ohmic losses are due to resistance through each of the PEMFC layers; the ohmic losses are removed by setting a high value for conductivity both in the membrane phase and solid phase. Kinetic losses occur due to the activation energy required for the electrochemical reactions. As a PEMFC operates, crossover gasses will permeate the membrane and react at the electrodes, leading to a mixed potential at the electrodes and an overall decrease in cell potential. The gas crossover effects lead to the difference between the thermodynamic potential and the open-circuit voltage. The thermodynamic potential, which is the maximum possible potential that can be achieved, is taken to be 1.18 A/cm^2 at 80°C.

5.7 Comparison of Dilute and Concentrated-Solution Theory

To illustrate the effects and needs for concentrated-solution theory, a comparison is made with a dilute-solution theory model using Nernst-Planck equation (5.36) for cerium

$$\mathbf{N}_{Ce} = -z_{Ce}u_{Ce}c_{Ce}F\nabla\Phi - \mathcal{D}_{Ce}\nabla c_{Ce} \tag{5.36}$$

where $\mathcal{D}_{Ce}\,(\lambda) = 3.11 \times 10^{-8}\,\lambda$ cm^2/s and $u_{Ce}\,(\lambda) = 1.89 \times 10^{-6}\lambda$ cm^2/V/s. The proton flux and water flux were calculated using Equations (5.17) and (5.19), where the $\nabla\mu_{Ce,H}$ terms are assumed to be zero. The diffusion and migration coefficients for cerium in Nafion as a function of water content are taken from Baker *et al.*[85] The dilute-solution theory model includes all of the chemical degradation reactions in Table 5-1 as well as the cerium effects on hydrogen activity. Polarization curves generated from the concentrated-solution-theory and dilute-solution-theory models are shown in Figure 5-6a & b.

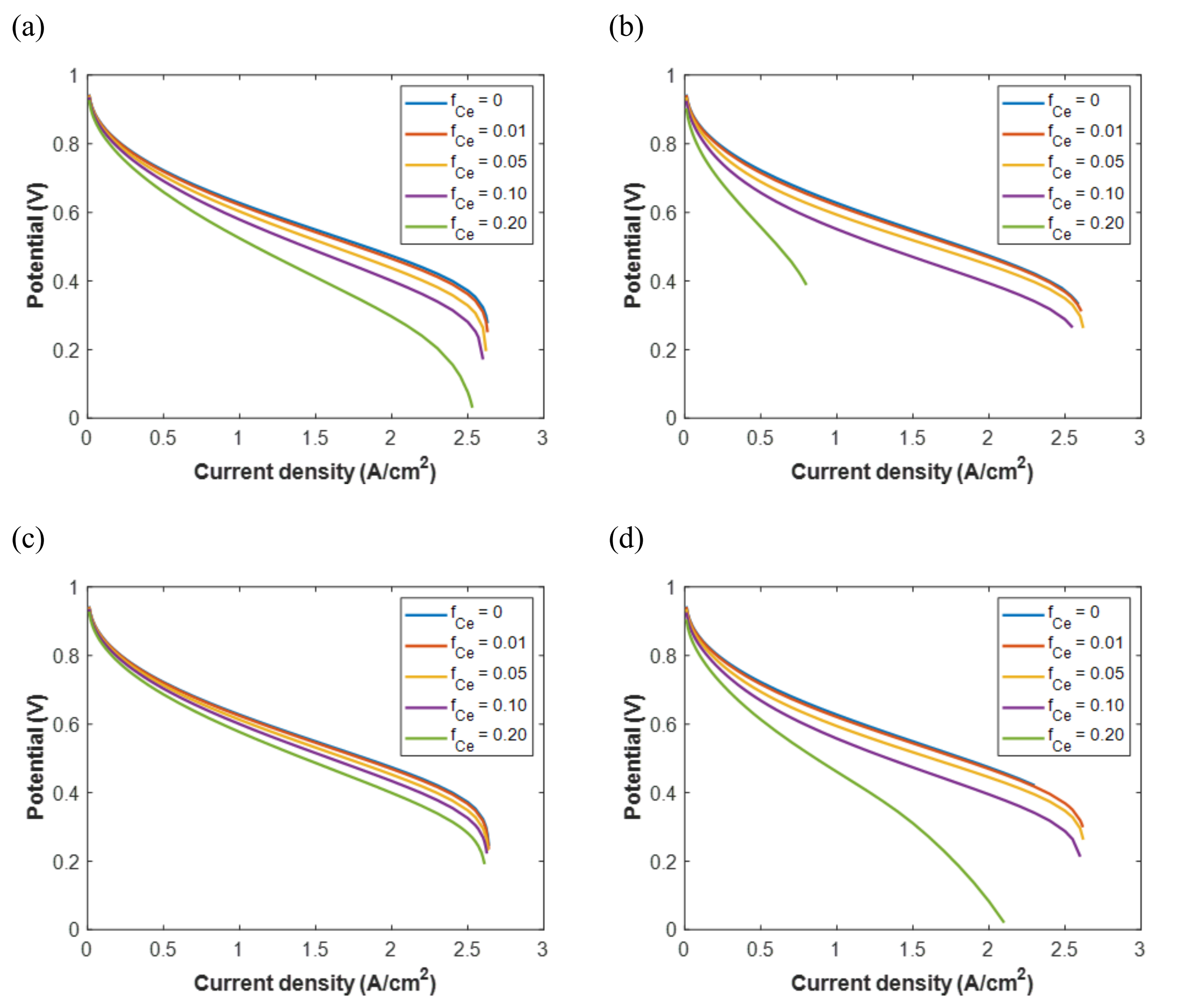


Figure 5-6: A comparison of polarization curves using various simulation approaches for cerium ion transport throughout the PEM, (a) concentrated-solution theory, (b) dilute-solution theory, (c) concentrated-solution theory without cerium-dependent properties, (d) dilute-solution theory without cerium-dependent properties. Simulation conditions are 80ºC, 1 bar, 90% RH, 100/60 standard cm^3/min air/H_2 flow rates.

As expected, the two models exhibit good agreement at low cerium concentrations. However, at higher cerium concentrations, the dilute-solution-theory model reaches mass-transport limitations at lower current densities as the cathode catalyst layer becomes saturated with cerium ions. The cerium content profiles in Figure 5-7 clearly demonstrate that as the current density increases, the potential gradient increases and the migration term drives cerium ions into the cathode catalyst layer. However, the migration term dominates the transport of cerium in the dilute-solution-theory model. Even at a low current density value of 0.01 A/cm^2, a concentration gradient of cerium across the PEMFC is predicted by the dilute-solution-theory model. In contrast, the concentrated-solution-theory model predicts a more uniform distribution of cerium across the cell at 0.01 A/cm^2. Therefore, the dilute-solution-theory model tends to overestimate the migration impact. The concentrated-solution-theory model corrects this term by including the solvent/ion interactions between the cerium ions and water, which drives cerium ions back toward the membrane and anode catalyst layer.

To analyze the impact of cerium on transport of water and protons through the membrane, both models are modified so that the transport properties (i.e. ξ, κ, $t_{Ce^{3+}}$, α_{ij}'s) are calculated for a membrane with zero cerium content. These polarization curves are shown in Figure 5-6c & d. The difference in the results in Figure 5-6a & c show that accounting for the amount of cerium in determining membrane transport properties has a significant effect in the concentrated-solution-theory model. The inclusion of cerium dependence leads to higher ohmic losses, as conductivity decreases with cerium content (see Figure 5-3). The limiting current density converges to the same value when the cerium effects on membrane transport properties are not considered; therefore, the increase in mass-transport limitations with cerium content can be attributed to the cerium effects on transport properties and not on the loss of proton activity due to the cerium ions. A comparison between the results in Figure 5-6b &d show the impact of cerium-ion effects on transport properties in the dilute-solution-theory model. There is little difference between the polarization curves at low cerium content due to the dominance of the cerium migration in comparison to the diffusion term. The limiting current density decreases with cerium content in both cases, with a steeper drop-off for the limiting current density when the cerium-dependent transport properties are used.

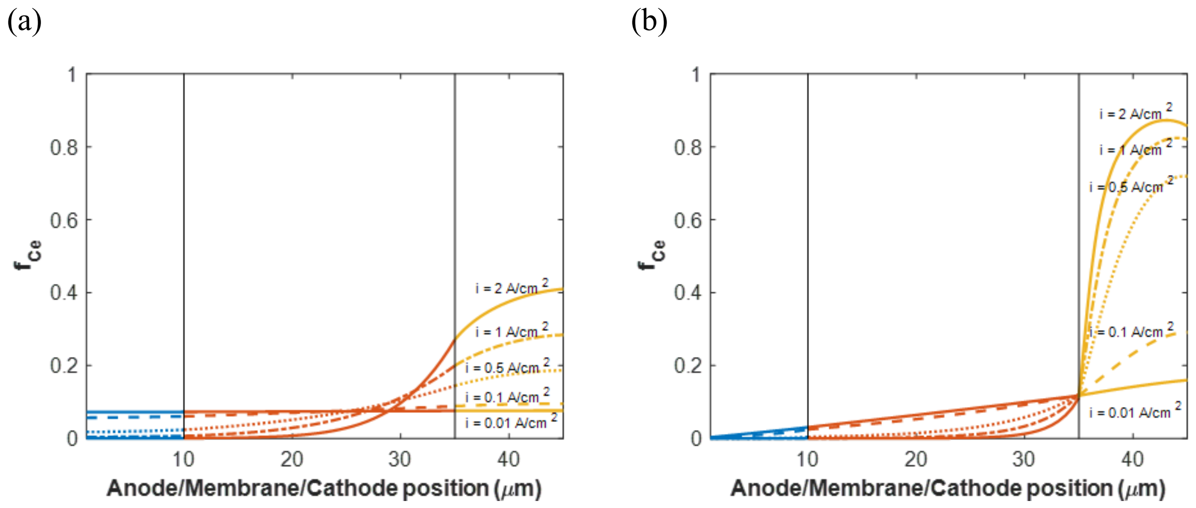


Figure 5-7: Concentration profiles for cerium based on a) concentrated-solution theory model and b) dilute-solution theory model. Simulation conditions are 80ºC, 1 bar, 90% RH, 1.67/1.0 cm^3/s air/feed flow rates, 10% f_{Ce}.

Figure 5-8 shows the dependence of cerium concentration throughout the cell on RH using the concentrated-solution-theory model. Increasing the RH into the cell drives the cerium ions back toward the anode catalyst layer and leads to better retention of cerium ions in the membrane, while decreasing the RH leads to accumulation of cerium in the cathode catalyst layer.

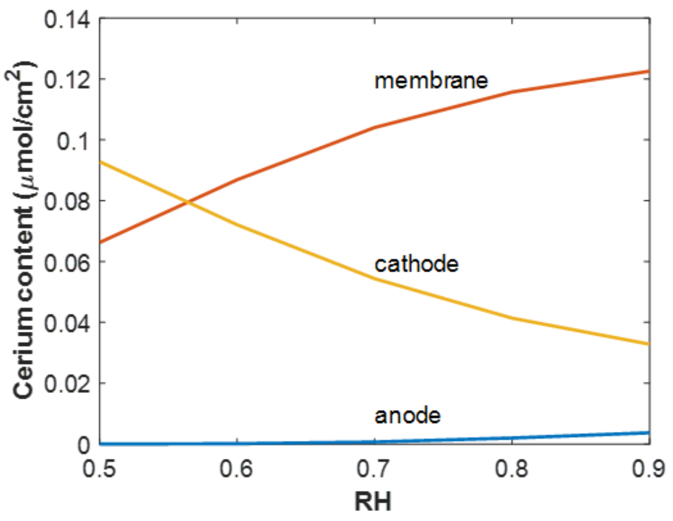


Figure 5-8: Cerium content as a function of RH at 0.5 A/cm^2. Simulation conditions are 80ºC, 1 bar, 1.67/1.0 cm^3/s air/feed flow rates, 10% f_{Ce}.

The results in Figure 5-9 provide a breakdown of the various driving forces for transport of water and cerium throughout the membrane. For both species, the migration term is positive, which means that the electrostatic forces are driving them from anode to cathode. At steady state, the migration term is balanced by the cerium and water electrochemical potential terms. The μ_{Ce}-driven term in the cerium flux increases in an exponential manner across the membrane, whereas the μ_w term in the water flux is roughly linear across the membrane. Thus, the primary driving force for the cerium ions entering the cathode is due to the Φ contribution to the overall flux, and the μ_{Ce} contribution drives cerium back toward the membrane and anode.

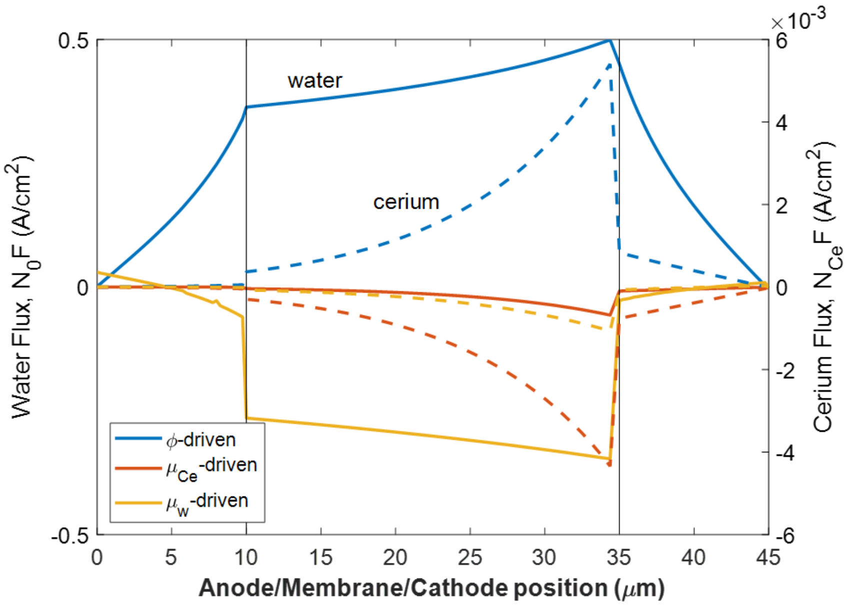


Figure 5-9: Driving forces for water flux (solid) and cerium flux (dashed) in the membrane. Positive flux is in the direction of anode to cathode. Simulation conditions are 80ºC, 1 bar, 100/60 sccm air/feed flow rates, 10% f_{Ce}, 90% RH and 0.5 A/cm^2.

5.8 Voltage-Loss Breakdown

To analyze the performance losses from the addition of cerium to the membrane, a voltage-loss-breakdown analysis was carried out for 5% f_{Ce} and 20% f_{Ce} in the membrane. The results in Figure 5-10 demonstrate that at low cerium content, below 5% f_{Ce}, the performance losses from the addition of cerium are small compared to the kinetic and ohmic losses. As the cerium content increases, the cerium-related voltage losses increase and become one of the primary sources of performance losses. The impact of cerium ions on mass-transport properties leads to the decrease in limiting current density in the PEMFC, whereas the presence of cerium ions in the catalyst-layer ionomer limiting access to catalyst sites contributes to losses in the ohmically limited region of the polarization curve. Both proton activity loss from cerium and cerium transport effects have significant contributions to the voltage loss, further amplifying the benefits of using a concentrated-solution-theory approach.

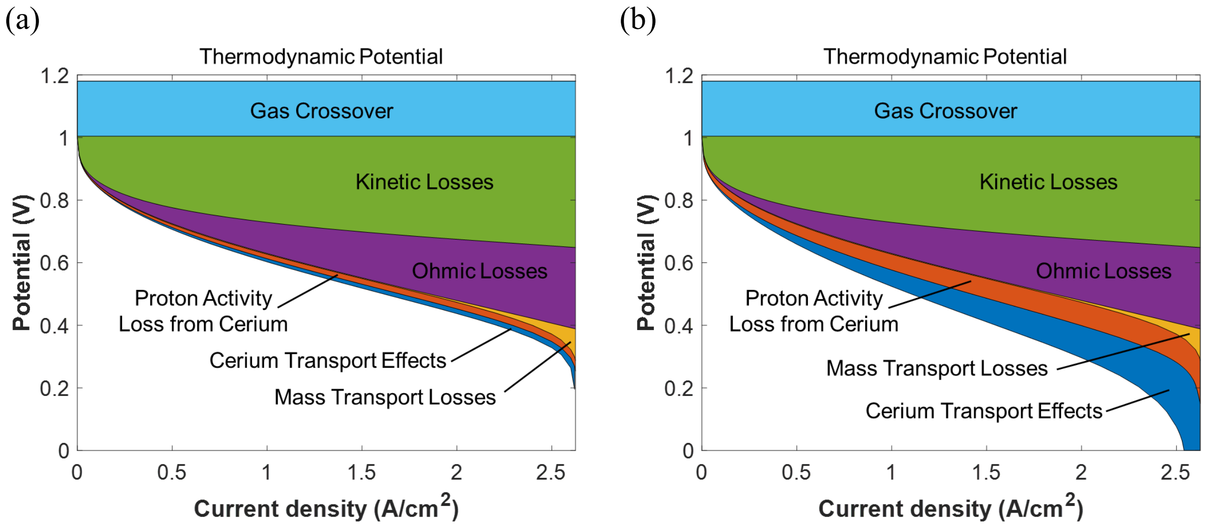


Figure 5-10: Voltage-loss breakdown for a cerium-doped membrane with (a) 5% f_{Ce} and (b) 20% f_{Ce}. Simulation conditions are 80ºC, 1 bar, 90% RH, 100/60 sccm air/feed flow rates.

5.9 Cerium Impacts on Durability and Performance

To look at the impact of cerium on the degradation rate, serval transient simulations were run at a constant current density.Figure 5-11 shows the effectiveness of adding cerium to the membrane in reducing FRR. Between 0% cerium content and 1% cerium content, the cumulative FRR decreases by two orders of magnitude. The FRR decreases further as more cerium is added, however the mitigation rate decreases; thereby suggesting that only minimal cerium is required. However, in full cells, multidimensional aspects and eventual cerium removal or interactions within the electrodes probably mean the values here are lower than those required in operation. While the cerium decreases the rate of chemical degradation in the membrane, the OCV also decreases. The OCV is highest at 0% cerium and decreases as more cerium is added to the membrane. However, the OCV decays over time when zero cerium is present in the membrane; in all cases with cerium, the OCV decay is negligible.

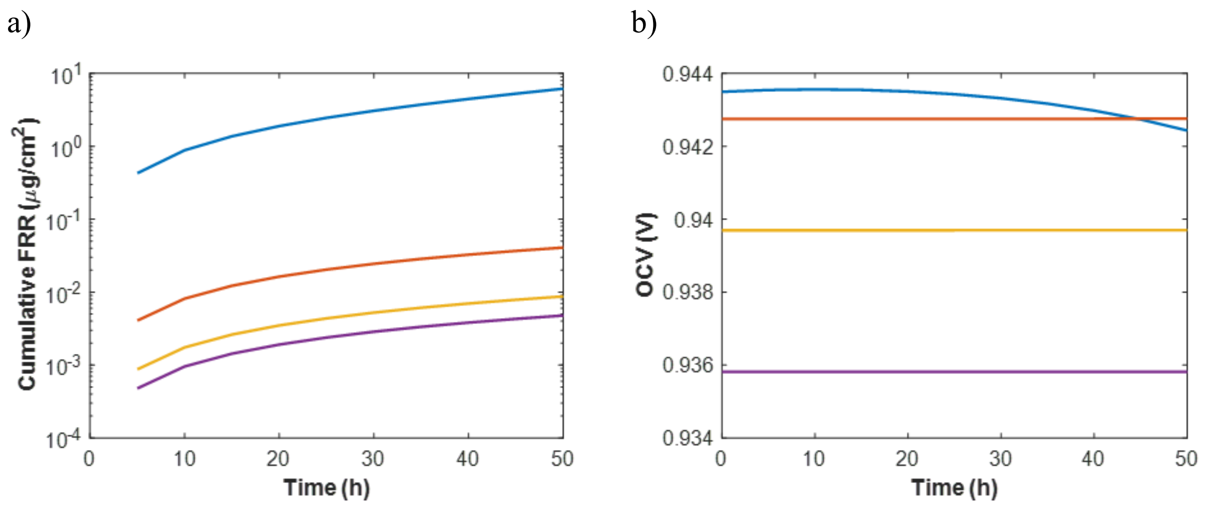


Figure 5-11: Transient simulation for various cerium concentrations, including a) cumulative FRR and b) OCV. Simulation conditions are 80ºC, 1 bar, 90% RH, 1.67/1.0 cm^3/s air/feed flow rates.

Performance and durability are often seen as competitive metrics; increasing membrane thickness is a strategy used to improve durability in commercial PEMFC vehicles, while decreasing membrane thickness is often the focus of research due to less material use, better water management and less ohmic drop, and thus a higher performance. Likewise, catalyst loading and subsequently catalyst-layer thickness is a critical design parameter. A sensitivity study was carried out to examine the effects of membrane thickness and catalyst-layer thickness on performance and durability as a function of cerium content. As was established by the results in Figure 5-11, the majority of the mitigation benefits occur with a cerium content of $f_{Ce} \leq 1\%$ in the membrane. The results in Figure 5-12 show the ratio of OCV to FRR for different membrane and catalyst-layer thicknesses and the respective times to failure, which is defined as a hydrogen crossover current density of > 2 mA/cm^2.[2]

Figure 5-12a & b demonstrates that the tradeoff between performance losses and degradation mitigation levels off very quickly, as the mitigation benefits increase quickly at small amounts of cerium and then begin to asymptote, whereas the OCV decrease with cerium content is more linear. The time to failure increases linearly with cerium content for all cases and increases with increasing membrane thickness and decreases with increasing catalyst-layer thickness. The thicker membranes increase lifetime because the crossover gases take more time to permeate the membrane, while the thicker catalyst layers decrease lifetime due to the increased reaction rate due to a greater availability of reaction sites for radical formation. These results show that an optimal tradeoff between performance and durability requires (within reason) thicker membranes and thinner catalyst layers, while also considering limitations due to local losses and flooding that are not included in this model.

(a)
(b)
(c)
(d)

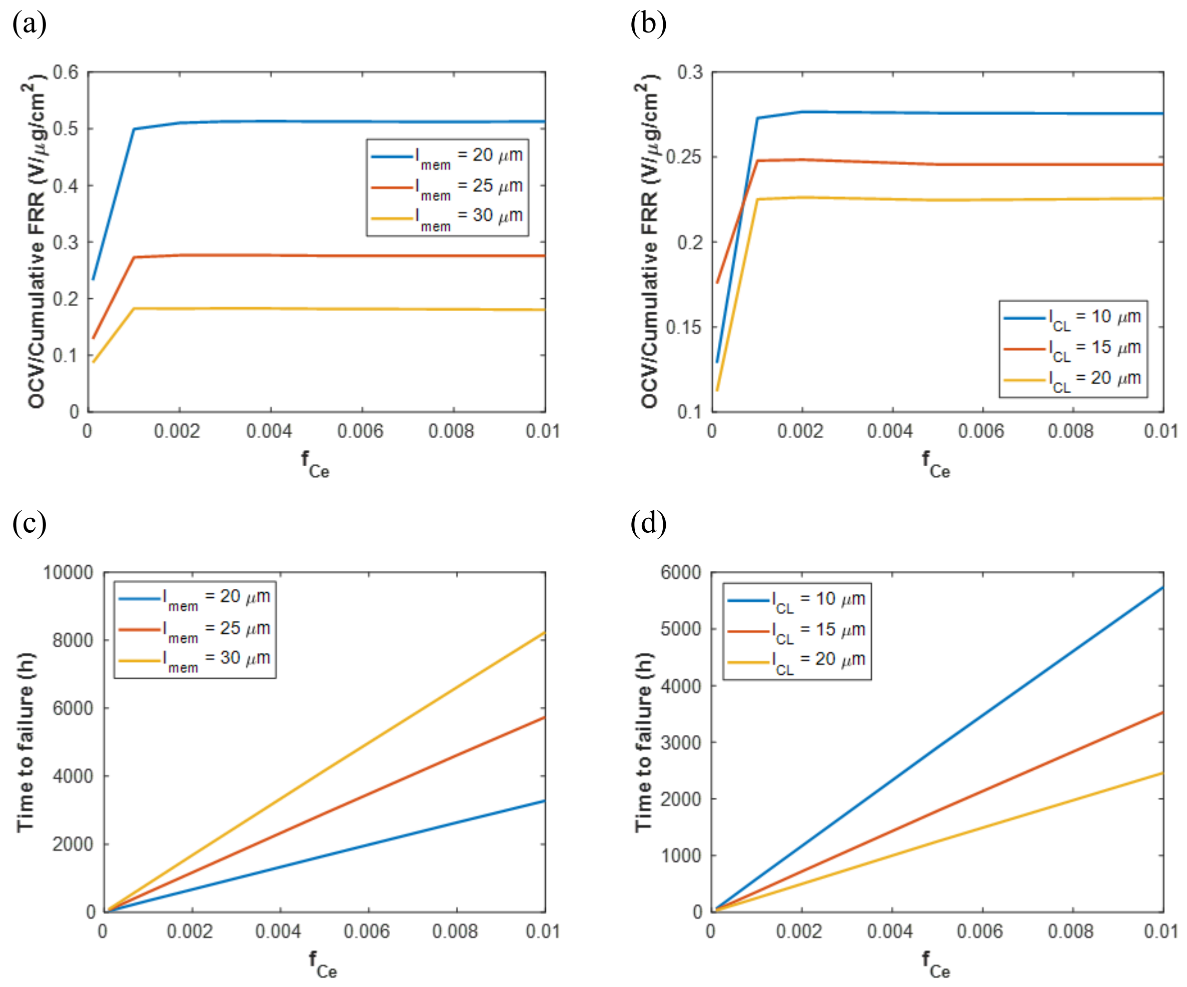


Figure 5-12: A comparison of performance and durability metrics for varying membrane thickness and catalyst-layer thickness: (a) varying membrane thickness with a constant catalyst-layer thickness (10 μm), (b) varying catalyst-layer thickness with a constant membrane thickness (25 μm). Time to failure as a function of cerium content for the cases in (a) and (b) are shown in (c) and (d). Simulation conditions are based on the DOE Membrane Chemical Durability Test: 90ºC, 1.5 bar, 30% RH, 0.23/0.63 cm^3/s air/feed flow rates.[2]

5.10 Effect of Cerium on Mechanical Degradation

In order to analyze the interactions between cerium transport and mechanical degradation phenomena, the micro-kinetic chemical degradation model with concentrated-solution-theory based transport of cerium ions was coupled with the mechanical degradation model described in Chapter 3. Simulations were run to show the impact of pinhole size on the distribution of cerium throughout the fuel cell. Figure 5-13 shows that increasing pinhole size leads to movement of cerium from the cathode back toward the membrane and anode. As the pinhole radius increases, the cerium content appears to approach a constant distribution. For higher current densities this distribution will have a higher cerium content in the cathode than at lower current densities. This is a result of larger pinholes causing conditions on the anode and cathode side of the membrane to approach equilibrium. In accordance with previous work, for small pinhole sizes ($\boldsymbol{r} \leq 500$ μm), a larger pinhole increases hydration throughout the fuel cell,[86] and higher water contents cause mass transport of cerium from the cathode into the membrane and anode ionomer.[87]

Further analysis was performed to determine the relationship between cerium loading and pinhole size and their effects on chemical degradation (i.e. FRR during operation). As has been extensively shown in experimental[2, 22, 26] and modeling studies,[35, 38, 39] increasing cerium in the membrane decreases FRR as shown in Figure 5-14. However, increasing pinhole size also decreases the FRR, as a result of having more cerium present in the anode and membrane with larger pinhole sizes. While increasing pinhole size leads to an increase in the gas crossover rate, this effect on the chemical degradation rate is counteracted by the more even distribution of cerium throughout the cell.

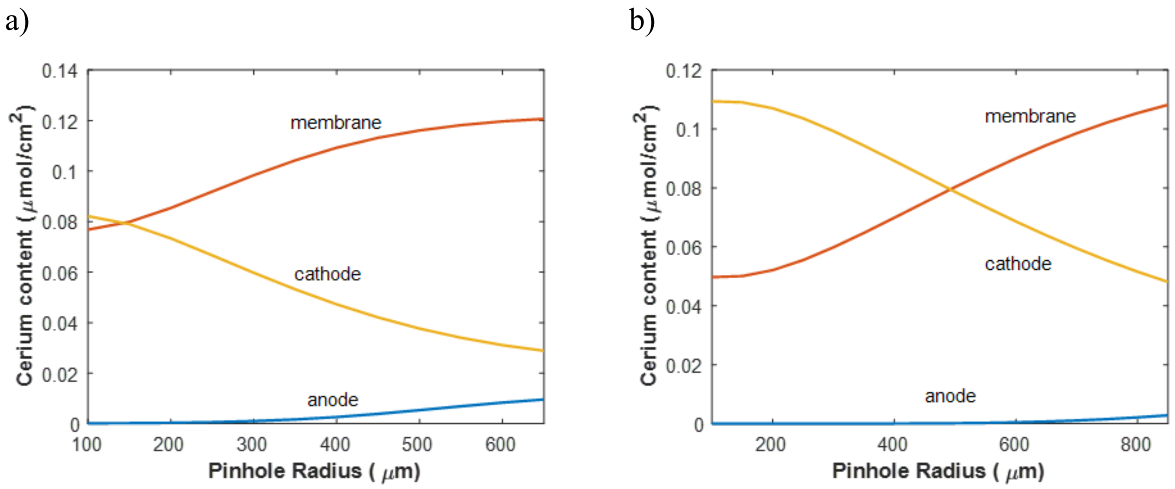


Figure 5-13: Effect of pinhole radius on cerium distribution in the MEA. Simulation conditions are 80°C, 1 bar, 10% f_{Ce}, 30% RH, air/feed rate 10/20 sccm, a) 0.1 A/cm^2 and b) 0.2 A/cm^2.

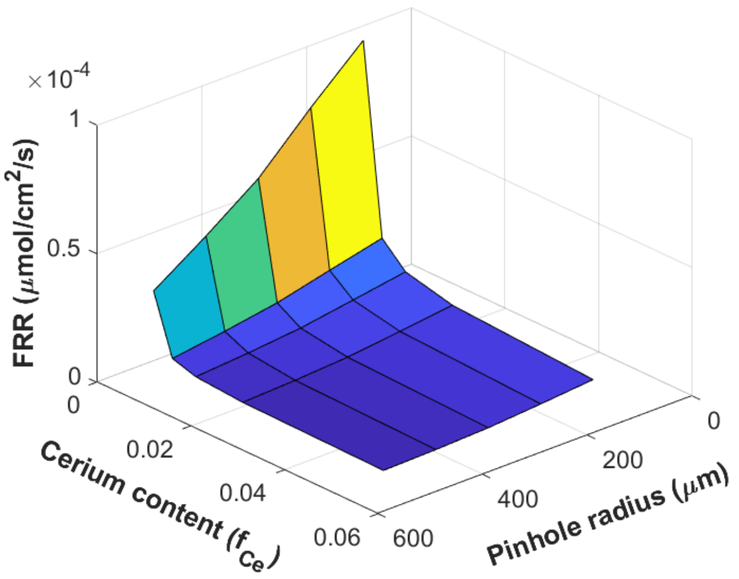


Figure 5-14: Effect of cerium content and pinhole radius on FRR. Simulation conditions are 80°C, 1 bar, 0.1 A/cm^2, 30% RH, air/feed rate 10/20 sccm.

To study the effects of cerium on mitigation of both chemical and mechanical degradation over time, simulations of humidity cycles and voltage cycles were carried out. The results of humidity cycling are shown in Figure 5-15 and the results of voltage cycling are shown in Figure 5-16.

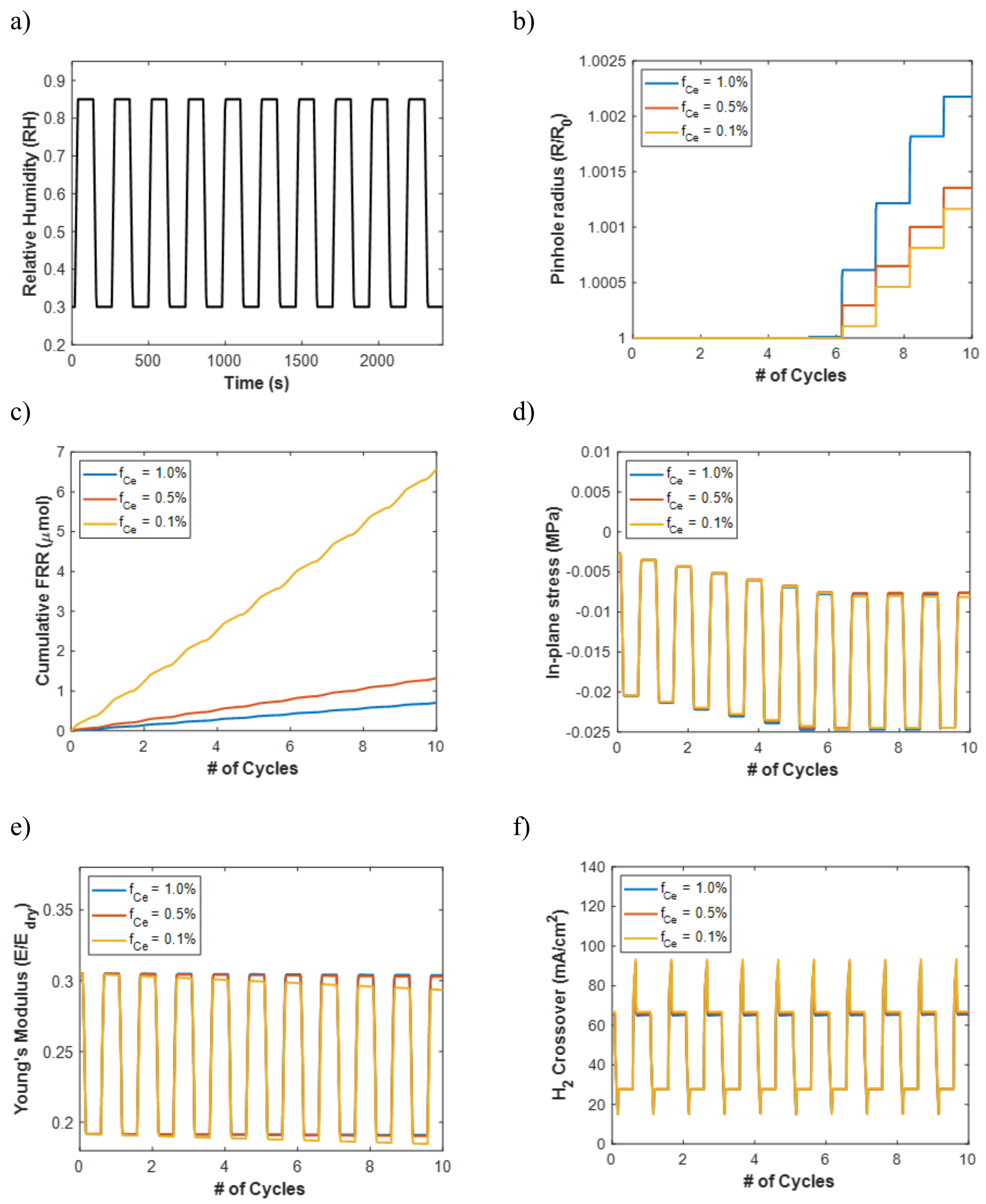
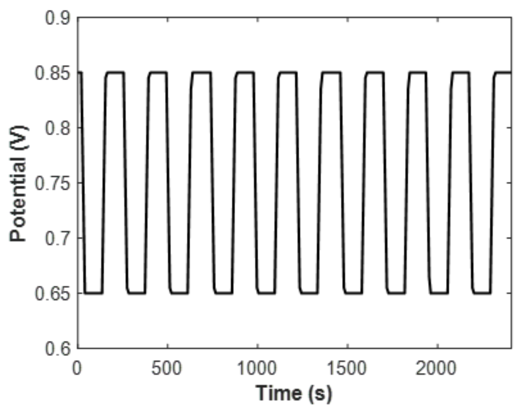


Figure 5-15: Simulation results of RH cycles at various cerium contents. a) Humidity cycles varying from 30% to 85% input into the model, b) pinhole growth rate with an initial pinhole radius $R_0 = 200$ μm, c) cumulative FRR in the cell summed over the cell cross-sectional area of 50 cm^2, d) in-plane stress normalized by Young's modulus (E_{dry}), e) normalized Young's modulus ($E_{\mathrm{dry}} = 250$ MPa), f) hydrogen crossover current density. Simulation conditions are 80ºC, 1 bar, 0.1 A/cm^2, air/feed stoichiometry 10/20.

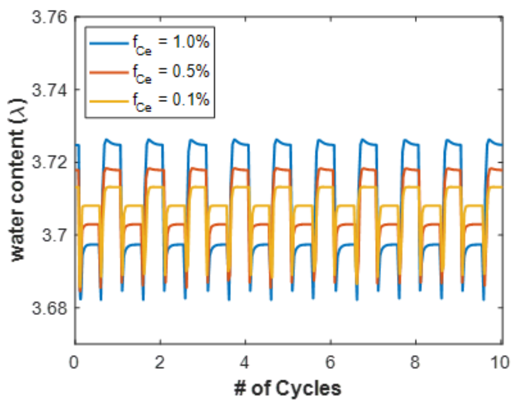
RH cycles from 30 to 85% were run, as shown in Figure 5-15a, with membrane cerium contents of 0.1, 0.5, and 1.0% and a pinhole with radius R_0 = 200 μm. The resulting pinhole-growth curves are given in Figure 5-15b and the FRR is shown in Figure 5-15c. The onset of plastic deformation does not occur until the sixth hydration cycle. This is because of the necessity to meet the plastic deformation condition in Equation (3.4), where the equivalent stress is equal to the yield strength. The Young's modulus decreases over time as the FRR increases, as reported by Kundu *et al.* [65] and Kusoglu *et al.* [67]. The in-plane stresses plotted in Figure 5-15d exhibit an initial decrease in stress for the first five hydration cycles before the deformation condition is met, and then the stresses remain fairly constant at the same point in the hydration cycle. The faster degradation at lower cerium content values causes the Young's modulus value to decrease, as shown in Figure 5-15e, which leads to a decrease in deformation. However, one should also account explicitly for the impact of cerium on Young's modulus and membrane properties in general, data that is incomplete in the literature. The cerium content does not have a significant impact on the hydrogen crossover rate, as shown in Figure 5-15f, as the gas crossover rate is dominated by transport through the pinhole, which is dictated by the Stefan-Maxwell equations and not dependent upon cerium concentration.

Voltage cycles from 0.85 to 0.65 V were run, as shown in Figure 5-16 a, with membrane cerium contents of 0.1, 0.5, and 1.0% and a pinhole with radius R_0 = 100 μm. The resulting water-content values are given Figure 5-16b. The water content increases with increasing cerium content, as demonstrated in Baker *et al.*[77] However, the change in water content over this voltage range is small in comparison to the RH cycling and is not large enough to cause any plastic deformation, so the pinhole radius does not change during the simulation. The results in Figure 5-16c show that the FRR decreases with increasing cerium content and increases at lower current densities, as there is a higher overpotential driving the electrochemical generation of hydrogen peroxide. The growth rate of FRR in the 0.01% cerium case decreases with time as the sulfonic-acid sites react with hydroxyl radicals. The more effective mitigation at 0.5 and 1.0% cerium leads to a more linear behavior in the FRR at the same point in the voltage cycle. Finally, the increase in cerium content causes an increase in hydrogen crossover current density, as shown in Figure 5-16d. With higher cerium content, the conductivity of the membrane decreases and subsequently the overall current density also decreases (see Figure 5-16e). As a result, more of the unreacted hydrogen is available to cross through the membrane through the pinhole, where it then reacts on the opposite electrode.

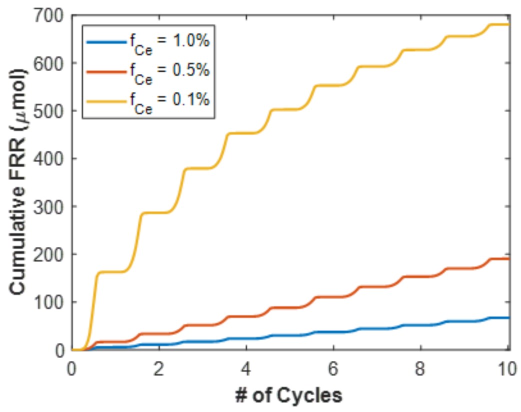
a)



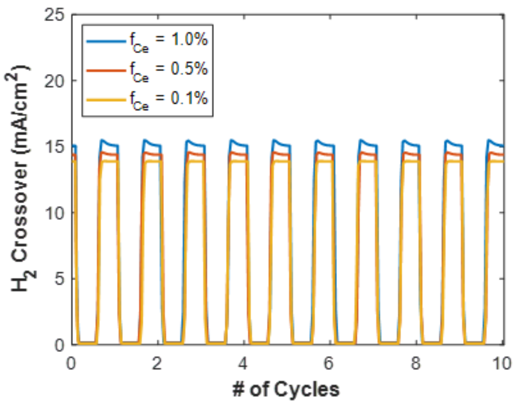
b)



c)



d)



e)

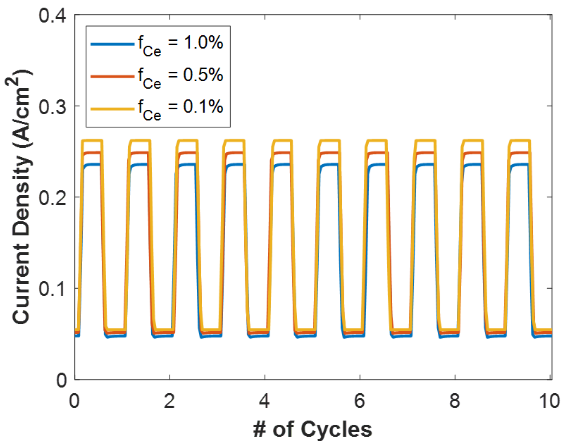


Figure 5-16: Simulation results of voltage cycles at various cerium contents. a) Voltage cycles varying from 0.85 V to 0.65 V input into the model, b) pinhole growth rate with an initial pinhole radius $\boldsymbol{R_0} = 100$ μm, c) cumulative FRR in the cell summed over the cell cross-sectional area of 50 cm^2, d) hydrogen crossover current density, e) current density. Simulation conditions are 80^{o}C, 1 bar, 30% RH at the anode and cathode, air/feed stoichiometry 10/20.

Chapter 6 – Approaches to Impedance Modeling of Porous Electrodes

Electrochemical-impedance-spectroscopy (EIS) techniques are frequently used to characterize the response of PEMFCS at the beginning-of-life and end-of-life conditions. These experiments can be useful for analyzing performance losses due to degradation phenomena. EIS experiments are carried out by applying a small sinusoidal perturbation in potential (or current) and measuring the current (or potential) response as a function of frequency of the input signal. The amplitude of the perturbation must be sufficiently small such that the response of the system is linear, which can be verified using the Kramers-Kronig relations.[88] The impedance spectra are typically fit to an equivalent circuit model consisting of resistive, capacitive, and inductive elements in order to deconvolute the limiting phenomena (e.g. kinetics, ohmic drop, mass-transport).[88-91] However, impedance spectra results may be difficult to interpret when the data can be fit to more than one equivalent circuit type. Another approach is to use a physics-based model to simulate the impedance response, which provides clearer distinction between which driving forces cause a certain impedance response. This section describes how a physics-based impedance model can be derived from the governing equations of a transient electrochemical model and compares several computational approaches.

6.1 Derivation of Impedance Model

The sinusoidal input to the impedance model can be expressed as the sum of a steady-state component and an oscillating component. According to Euler's law, the oscillating component may be written as a complex number (note that the convention $j = \sqrt{-1}$ is used, consistent with electrical engineering literature where i is used to denote current),

$$e^{jt} = \cos t + j \sin t \tag{6.1}$$

Likewise, each dependent variable $x(t)$ is expressed as a sum of a steady-state part plus a small perturbation,[92-94]

$$x(t) = \bar{x} + \mathrm{Re}\{\tilde{x} \exp(j\omega t)\} \tag{6.2}$$

where $\bar{x}$ is the steady state component of $x(t)$, $\tilde{x}$ is the oscillating component of $x(t)$ and is a complex number, and ω is the angular frequency. To construct an impedance model, the governing equations in the time domain must be transformed to the frequency domain by representing the governing equations as phasors or using a Laplace transform:

$$\tilde{x}(j\omega) = \int_0^\infty x(t) e^{-j\omega t} dt \tag{6.3}$$

If the governing equations in the time domain are nonlinear, they must be linearized around the steady-state value with higher-order terms disregarded.[95]

$$f\big(x(t)\big) = f\big(\bar{x} + Re\{\tilde{x}e^{j\omega t}\}\big) = f(\bar{x}) + \frac{df}{dx}\bigg|_{\bar{x}} Re\{\tilde{x}e^{j\omega t}\} \tag{6.4}$$

The impedance response of a system may be extracted from the frequency domain model and is defined as the transfer function between the potential and the current.

$$Z(\omega) = \frac{\tilde{\Phi}(\omega)}{\tilde{\imath}(\omega)} \tag{6.5}$$

In order to measure or simulate the full impedance response, a range of frequencies should be used such that the imaginary component of the impedance approaches zero at both the high and low ends of the frequency range.[88] Additionally, the magnitude of the perturbation used to probe the system must be sufficiently small such that the system response remains linear, but also large enough to distinguish it from noise. The magnitude of this perturbation varies depending on how nonlinear the system behavior is; linear systems may use a large amplitude while highly nonlinear systems will require a very small amplitude.[88]

In highly coupled, nonlinear systems such as a PEMFC, linearization of the model may be impractical without making significant simplifying assumptions to the model. In these cases, the development of an algorithm to numerically linearize the governing equations would be advantageous and is a suggested topic of future work in the community.

Three approaches to simulating the impedance response numerically are described and compared here. The first is a transient simulation of a sine wave of small amplitude and numerical integration to obtain impedance. This approach is modeled after the experimental techniques for EIS. The frequency response of the system can be obtained from the time-domain signals using a Fourier transform,

$$\tilde{\imath}_{Re}(\omega) = \frac{1}{T}\int_0^T i(t)\cos(\omega t)\,dt \tag{6.6}$$

$$\tilde{\imath}_{Im}(\omega) = -\frac{1}{T}\int_0^T i(t)\sin(\omega t)\,dt \tag{6.7}$$

$$\tilde{\Phi}_{Re}(\omega) = \frac{1}{T}\int_0^T \Phi(t)\cos(\omega t)\,dt \tag{6.8}$$

$$\tilde{\Phi}_{Im}(\omega) = -\frac{1}{T}\int_0^T \Phi(t)\sin(\omega t)\,dt \tag{6.9}$$

where T is the period of an integer number of cycles at frequency ω.[88] In a numerical simulation these equations are evaluated using a numerical integration method, such as the Trapezoidal rule. This approach is the most computationally intensive but requires no additional development from the transient electrochemical model. This approach has been used in a few modelling studies where the governing equations are coupled and non-linear.[96, 97] This method is also constrained by the limitations of machine precision, as the magnitude of the perturbation required to maintain linearity may be below the typical double precision error (~10^{-16}). Quadratic precision or higher may be used in these cases if supported by the programming language used. Additionally, the accuracy of this computation is dependent upon the magnitude of the time step used in the transient simulation, so an accurate simulation may require more time steps and therefore a much longer total computation time.

The second approach to modeling impedance is the transformation of the time-domain model equations into frequency-domain model equations using Laplace transforms and linearized about the steady-state model solution (see Equation (6.4)). The third approach builds on the second approach by splitting up the frequency domain model equations into real and imaginary components such that the total number of equations is doubled. Two sets of governing equations are written, one for the real components of each variable and one for the imaginary components of each variable. This approach takes advantage of the Cauchy-Riemann equations, which state that for a complex variable $z = x + jy$ and $f(z) = u(x, y) + jv(x, y)$,

$$\frac{du}{dx} = \frac{dv}{dy} \tag{6.10}$$

$$\frac{du}{dy} = -\frac{dv}{dx} \tag{6.11}$$

These equations allow the derivatives of complex variables to be written in terms of their real and imaginary components. The third approach is used when complex numbers are disallowed in the programming language used.[42, 94, 98-100] Modern programming languages such as MATLAB, Python, and C include a complex number data type, therefore the second approach is easier to implement and splitting up equations into real and imaginary components is unnecessary. Additionally, several models have been developed that may be solved analytically.[92, 93, 95, 101-106] This approach can only be used on systems of equations that can be solved analytically, which limits the complexity of the model and may require several simplifying assumptions.

Two case studies are presented in order to further illustrate and compare the three approaches for numerically simulating impedance. The first case study consists of a 1D porous electrode with linear kinetics and uniform concentration, as described in Newman and Tobias.[107] The second case study is a simple PEMFC cathode catalyst layer model, which includes the ORR with Tafel kinetics and oxygen diffusion using Fick's law, as described by Kulikovsky.[108]

6.2 Case Study: Porous Electrode with Linear Kinetics

The governing equations for the time domain are listed in Table 6-1. The system variables include the solid-phase current density (i_1), the electrolyte-phase current density (i_2), the solid-phase potential (Φ_1), and the electrolyte-phase potential (Φ_2). The governing equations consistent of Ohm's law for the solid-phase current and the electrolyte-phase current, a conservation of charge equation, and a charge balance including the reaction term. Since the concentration is uniform, the kinetics are independent of concentration. The boundary conditions set a cell potential at $x = L$, which contains the oscillating input to the simulation:

$$\Phi_1(t) = \Phi_{cell} + \Delta\Phi \cos(\omega t) \tag{6.12}$$

At $x = 0$, all of the current is carried it the electrolyte phase and at $x = L$, all of the current is transferred into the solid phase.

Table 6-1: Case Study 1 Time Domain Equations

$\boldsymbol{x = 0}$			$\boldsymbol{x = L}$
	1	$i_1 = -\sigma \dfrac{d\Phi_1}{dx}$	$\Phi_1 = \Phi_{cell} + \Delta\Phi \cos(\omega t)$
	2	$i_2 = -\kappa \dfrac{d\Phi_2}{dx}$	$i_2 = 0$
$i_1 = 0$	3	$\dfrac{di_1}{dx} + \dfrac{di_2}{dx} = 0$	
$\Phi_2 = 0$	4	$\dfrac{di_2}{dx} = ai_0 \dfrac{\alpha_c F}{RT}(\Phi_1 - \Phi_2) + aC\dfrac{d(\Phi_1 - \Phi_2)}{dt}$	

To transform the time-domain governing equations into the frequency domain, each equation and boundary condition are represented as phasors. In this case, all of the equations are already linear, therefore taking the Laplace transform is equivalent to substituting Equation (6.2) into the time-domain equations and separating the steady-state and oscillating components. For example, using the first equation in Table 6-1,

$$i_1(t) = -\sigma \frac{d\Phi_1(t)}{dx} \tag{6.13}$$

$$\bar{i}_1 + Re\{\tilde{i}_1 e^{j\omega t}\} = -\sigma \frac{d\left(\bar{\Phi}_1 + Re\{\tilde{\Phi}_1 e^{j\omega t}\}\right)}{dx} \tag{6.14}$$

The steady-state solution can be separated, leaving only the oscillating terms:

$$\left(\bar{i}_1 + \sigma \frac{d\bar{\Phi}_1}{dx}\right) + Re\{\tilde{i}_1 e^{j\omega t}\} = -\sigma \frac{d\left(Re\{\tilde{\Phi}_1 e^{j\omega t}\}\right)}{dx} \tag{6.15}$$

$$Re\{\tilde{i}_1 e^{j\omega t}\} = -\sigma \frac{d\left(Re\{\tilde{\Phi}_1 e^{j\omega t}\}\right)}{dx} \tag{6.16}$$

In the time domain, only the real component of the oscillating signal is observed, but once transformed into the frequency domain, the oscillating component includes both the real and imaginary parts. Equation (6.16) can be rewritten as

$$\tilde{i}_1 e^{j\omega t} = -\sigma \frac{d\left(\tilde{\Phi}_1 e^{j\omega t}\right)}{dx} \tag{6.17}$$

The $e^{j\omega t}$ terms on both sides of the equation cancel, which leaves

$$\tilde{\imath}_1 = -\sigma \frac{d\tilde{\Phi}_1}{dx} \tag{6.18}$$

The same procedure may be carried out for Equations 2-4 in Table 6-1. The last equation contains a time derivative that must be evaluated.

$$\frac{d\tilde{\imath}_2 e^{j\omega t}}{dx} = ai_0 \frac{\alpha_c F}{RT}\left(\tilde{\Phi}_1 e^{j\omega t} - \tilde{\Phi}_2 e^{j\omega t}\right) + aC\frac{d\left(\tilde{\Phi}_1 e^{j\omega t} - \tilde{\Phi}_2 e^{j\omega t}\right)}{dt} \tag{6.19}$$

$$\frac{d\tilde{\imath}_2 e^{j\omega t}}{dx} = ai_0 \frac{\alpha_c F}{RT}\left(\tilde{\Phi}_1 e^{j\omega t} - \tilde{\Phi}_2 e^{j\omega t}\right) + aC\left(\tilde{\Phi}_1 e^{j\omega t}\right)(j\omega) - aC\left(\tilde{\Phi}_2 e^{j\omega t}\right)(j\omega) \tag{6.20}$$

$$\frac{d\tilde{\imath}_2}{dx} = ai_0 \frac{\alpha_c F}{RT}\left(\tilde{\Phi}_1 - \tilde{\Phi}_2\right) + j\omega aC\left(\tilde{\Phi}_1 - \tilde{\Phi}_2\right) \tag{6.21}$$

Steady-state Dirichlet boundary conditions can be set to zero in the frequency domain as the variable is constant over time, whereas oscillating Dirichlet boundary conditions must be specified. The boundary condition for Φ_1 can be transformed using $\overline{\Phi}_1 = \Phi_{cell}$ and Euler's identity in Equation (6.1):

$$\overline{\Phi}_1 + Re\left\{\tilde{\Phi}_1 e^{j\omega t}\right\} = \Phi_{cell} + \Delta\Phi\cos(\omega t) \tag{6.22}$$

$$Re\left\{\tilde{\Phi}_1 e^{j\omega t}\right\} = \Delta\Phi\cos(\omega t) \tag{6.23}$$

$$Re\left\{\tilde{\Phi}_1 e^{j\omega t}\right\} = Re\left\{\Delta\Phi e^{j\omega t}\right\} \tag{6.24}$$

$$\tilde{\Phi}_1 = \Delta\Phi \tag{6.25}$$

The transformed set of governing equations in the frequency domain are listed in Table 6-2.

Table 6-2: Case Study 1 Frequency Domain Equations

$\boldsymbol{x = 0}$			$\boldsymbol{x = L}$
	1	$\tilde{\imath}_1 = -\sigma \frac{d\tilde{\Phi}_1}{dx}$	$\tilde{\Phi}_1 = \Delta\Phi$
	2	$\tilde{\imath}_2 = -\kappa \frac{d\tilde{\Phi}_2}{dx}$	$\tilde{\imath}_2 = 0$
$\tilde{\imath}_1 = 0$	3	$\frac{d\tilde{\imath}_1}{dx} + \frac{d\tilde{\imath}_2}{dx} = 0$	
$\tilde{\Phi}_2 = 0$	4	$\frac{d\tilde{\imath}_2}{dx} = ai_0 \frac{\alpha_c F}{RT}\left(\tilde{\Phi}_1 - \tilde{\Phi}_2\right) + j\omega aC\left(\tilde{\Phi}_1 - \tilde{\Phi}_2\right)$	

Finally, the frequency-domain equations can be separated into their real and imaginary components. Each of the equations and boundary conditions in Table 6-2 can be written as two equations, one for the real component and one for the imaginary component. For Equation 1 in Table 6-2 this results with the two equations,

$$\tilde{i}_{1,Re} = -\sigma \frac{d\widetilde{\Phi}_{1,Re}}{dx} \tag{6.26}$$

$$\tilde{i}_{1,Im} = -\sigma \frac{d\widetilde{\Phi}_{1,Im}}{dx} \tag{6.27}$$

Likewise, this procedure can be carried out for Equations 2-4 in Table 6-2. However, Equation 4 contains a term multiplied by $(j\omega)$ which must be eliminated to solve the system of equations in terms of real numbers only.

$$\frac{d\tilde{i}_{2,Re}}{dx} = ai_0 \frac{\alpha_c F}{RT}\left(\widetilde{\Phi}_{1,Re} - \widetilde{\Phi}_{2,Re}\right) + j\omega aC\left(\widetilde{\Phi}_{1,Re} - \widetilde{\Phi}_{2.Re}\right) \tag{6.28}$$

$$\frac{d\tilde{i}_{2,Im}}{dx} = ai_0 \frac{\alpha_c F}{RT}\left(\widetilde{\Phi}_{1,Im} - \widetilde{\Phi}_{2,Im}\right) + j\omega aC\left(\widetilde{\Phi}_{1,Im} - \widetilde{\Phi}_{2,Im}\right) \tag{6.29}$$

Using the Cauchy-Reimann equations, the imaginary terms can be substituted as $j\omega\widetilde{\Phi}_{1,Re} = -\omega\widetilde{\Phi}_{1,Im}$ and $j\omega\widetilde{\Phi}_{1,Im} = \omega\widetilde{\Phi}_{1,Re}$ (likewise for $\widetilde{\Phi}_{2,Re}$ and $\widetilde{\Phi}_{2,Im}$).

$$\frac{d\tilde{i}_{2,Re}}{dx} = ai_0 \frac{\alpha_c F}{RT}\left(\widetilde{\Phi}_{1,Re} - \widetilde{\Phi}_{2,Re}\right) - \omega aC\left(\widetilde{\Phi}_{1,Im} - \widetilde{\Phi}_{2.Im}\right) \tag{6.30}$$

$$\frac{d\tilde{i}_{2,Im}}{dx} = ai_0 \frac{\alpha_c F}{RT}\left(\widetilde{\Phi}_{1,Im} - \widetilde{\Phi}_{2,Im}\right) + \omega aC\left(\widetilde{\Phi}_{1,Re} - \widetilde{\Phi}_{2,Re}\right) \tag{6.31}$$

As was shown in Equation (6.24), the oscillating boundary condition for $\widetilde{\Phi}_1$ contains the real part of the variable $\left(\widetilde{\Phi}_{1,Re} = \Delta\Phi\right)$ and represents the reference phase of the cell, which can be represented by $\widetilde{\Phi}_{1,Im} = 0$. The complete set of equations and boundary conditions split into real and imaginary parts are listed in Table 6-3.

Simulations were carried out using each of the three approaches described, and the corresponding MATLAB codes can be found in Appendix B.5. The impedance for this system was calculated as

$$Z(\omega) = \left.\frac{\widetilde{\Phi}_1(\omega)}{\tilde{i}_1(\omega)}\right|_{x=L} \tag{6.32}$$

The resulting Nyquist plots of the impedance spectra are shown in Figure 6-1. The results for Approaches 2 and 3 match exactly. The results from approach 1 have a slightly lower amplitude, but the shape of the impedance spectra, known as the "kinetic loop," and distribution of frequency matches fairly well. Approach 2 had a CPU time of 1.7 seconds and Approach 3 had a CPU time of 1.1 seconds. In comparison, Approach 1 was much slower, with a CPU time ranging from 130 to 1370 seconds with a step size of 100 and 1000 points per period, respectively.

Table 6-3: Case Study 1 Real and Imaginary Frequency Domain Equations

$x = 0$			$x = L$
	1	$\tilde{i}_{1,Re} = -\sigma \frac{d\widetilde{\Phi}_{1,Re}}{dx}$	$\widetilde{\Phi}_{1,Re} = \Delta\Phi$
	2	$\tilde{i}_{2,Re} = -\kappa \frac{d\widetilde{\Phi}_{2,Re}}{dx}$	$\tilde{i}_{2,Re} = 0$
$\tilde{i}_{1,Re} = 0$	3	$\frac{d\tilde{i}_{1,Re}}{dx} + \frac{d\tilde{i}_{2,Re}}{dx} = 0$	
$\widetilde{\Phi}_{2,Re} = 0$	4	$\frac{d\tilde{i}_{2,Re}}{dx} = ai_0 \frac{\alpha_c F}{RT}(\widetilde{\Phi}_{1,Re} - \widetilde{\Phi}_{2,Re}) - \omega aC(\widetilde{\Phi}_{1,Im} - \widetilde{\Phi}_{2,Im})$	
	5	$\tilde{i}_{1,Im} = -\sigma \frac{d\widetilde{\Phi}_{1,Im}}{dx}$	$\widetilde{\Phi}_{1,Im} = 0$
	6	$\tilde{i}_{2,Im} = -\kappa \frac{d\widetilde{\Phi}_{2,Im}}{dx}$	$\tilde{i}_{2,Im} = 0$
$\tilde{i}_{1,Im} = 0$	7	$\frac{d\tilde{i}_{1,Im}}{dx} + \frac{d\tilde{i}_{2,Im}}{dx} = 0$	
$\widetilde{N}_{O_2,Im} = 0$	8	$\frac{d\tilde{i}_{2,Im}}{dx} = ai_0 \frac{\alpha_c F}{RT}(\widetilde{\Phi}_{1,Im} - \widetilde{\Phi}_{2,Im}) + \omega aC(\widetilde{\Phi}_{1,Re} - \widetilde{\Phi}_{2,Re})$	

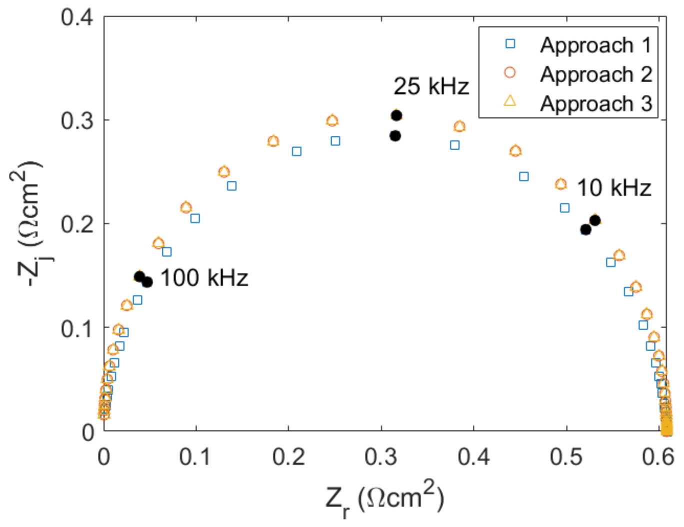


Figure 6-1: Comparison of impedance model simulation techniques for a porous electrode with linear kinetics. Approach 1 uses 1000 points per period.

To further illustrate the importance of the time discretization parameter for Approach 1, simulation results using several different time step sizes are compared in Figure 6-2. Approximately 1500 mesh points per period were necessary in order to reach a mesh independent solution. However, this solution is still offset from the solution calculated using Approaches 2 and 3. While more time points result in a more calculation, it also requires increased computation time, which is a major limitation of this approach.

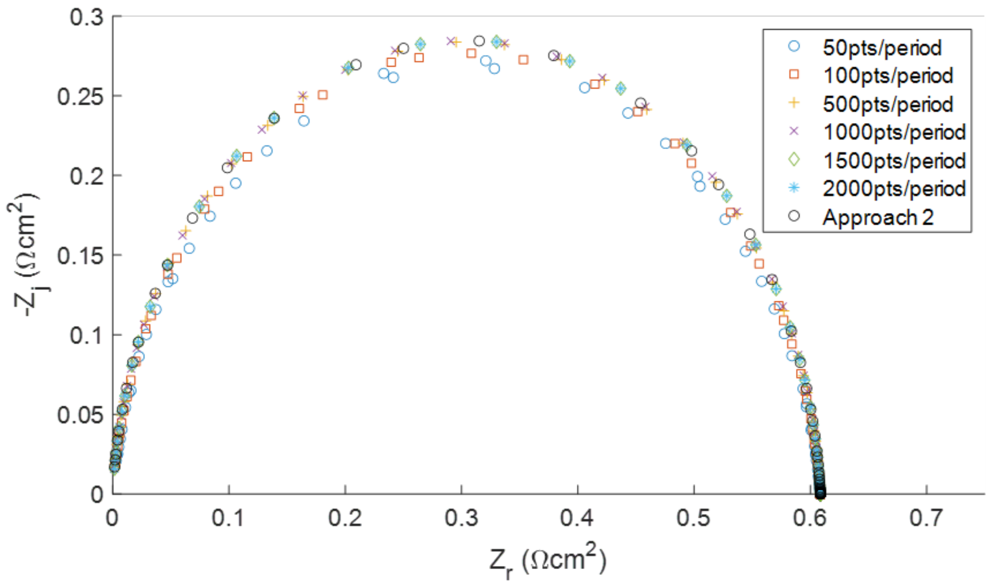


Figure 6-2: Impedance spectra calculated by Approach 1 using various time domain mesh sizes.

Additionally, a nonlinear regression was carried out on a R-RCPE equivalent circuit for the three approaches,

$$Z = R_e + \frac{R_t}{1 + (j\omega R_t C_{dl})^{\alpha}} \tag{6.33}$$

where R_e is the ohmic resistance, R_t is the charge transfer resistance, C_{dl} is the double layer capacitance, and α is a parameter associated with constant-phase element (CPE) behavior. The results are listed in Table 6-4. The fit to a R-RCPE model for Approaches 2 and 3 show that the model reduces to purely RC (resistive-capacitive) and the results are Kramers-Kronig consistent. However, the results from Approach 1 are not Kramers-Kronig consistent, and these inconsistencies that the results have numerical artifacts throughout the simulation range in the imaginary part.

Table 6-4: Nonlienar Regression of R-RCPE Equivalent Circuits

	Approach 1	**Approach 2**	**Approach 3**
R_e	8.10×10^{-4}	0	0
R_t	0.606	0.609	0.609
C_{dl}	1.01×10^{-5}	1.00×10^{-5}	1.00×10^{-5}
α	0.976	1	1

6.3 Case Study: Porous Electrode with Diffusion of Reactant Species and Tafel Kinetics

The governing equations for the time domain are listed in Table **6-5**. The first three equations are the same as in the first case study. In Equation 4 Tafel kinetics are used instead of linear kinetics, and therefore the equation is no longer linear. Two additional equations are added for the flux and concentration of oxygen. The concentration of oxygen is dictated by Fick's law and the flux of oxygen is determined by a mass balance with a Tafel kinetics source term. By using first-order equations, the derivation of impedance equations is simplified as there are no higher-order terms to consider.

Table 6-5: Case Study 2 Time Domain Equations

$\boldsymbol{x = 0}$			$\boldsymbol{x = L}$
	1	$i_1 = -\sigma \frac{d\Phi_1}{dx}$	$\Phi_1 = \Phi_{cell} + \Delta\Phi \cos(\omega t)$
	2	$i_2 = -\kappa \frac{d\Phi_2}{dx}$	$i_2 = 0$
$i_1 = 0$	3	$\frac{di_1}{dx} + \frac{di_2}{dx} = 0$	
$\Phi_2 = 0$	4	$\frac{di_2}{dx} = -ai_0 \left(\frac{py_{O_2}}{p^{ref}}\right) \exp\left(-\frac{\alpha_c F}{RT}(\Phi_1 - \Phi_2 - U^0)\right) + aC \frac{d(\Phi_1 - \Phi_2)}{dt}$	
$N_{O_2} = 0$	5	$N_{O_2} = -D_{O_2,w} \frac{p}{RT} \frac{dy_{O_2}}{dx}$	
	6	$\frac{p}{RT} \frac{dy_{O_2}}{dt} = \frac{dN_{O_2}}{dx} + \frac{ai_0}{4F} \left(\frac{py_{O_2}}{p^{ref}}\right) \exp\left(-\frac{\alpha_c F}{RT}(\Phi_1 - \Phi_2 - U^0)\right)$	$y_{O_2} = 0.21\left(1 - \frac{RH}{p_w^{sat}}\right)$

The same procedure as the previous case study can be applied to Equations 1-3 and 5. However, since Equations 4 and 6 are nonlinear, they must first be linearized around the steady-state value. For Equation 4,

$$f(i_2, y_{O_2}, \Phi_1, \Phi_2) = \frac{di_2}{dx} + ai_0 \left(\frac{py_{O_2}}{p^{ref}}\right) \exp\left(-\frac{\alpha_c F}{RT}(\Phi_1 - \Phi_2 - U^0)\right) + aC \frac{d(\Phi_1 - \Phi_2)}{dt} \quad (6.34)$$

$$f(i_2, y_{O_2}, \Phi_1, \Phi_2) = \left.\frac{df}{di_2}\right|_{\bar{y}_{O_2}, \bar{\Phi}_1, \bar{\Phi}_2} + \left.\frac{df}{dy_{O_2}}\right|_{\bar{\imath}_2, \bar{\Phi}_1, \bar{\Phi}_2} + \left.\frac{df}{d\Phi_1}\right|_{\bar{\imath}_2, \bar{y}_{O_2}, \bar{\Phi}_2} + \left.\frac{df}{\mathrm{d}\Phi_2}\right|_{\bar{\imath}_2, \bar{y}_{O_2}, \bar{\Phi}_1} \quad (6.35)$$

$$f\left(i_2, y_{O_2}, \Phi_1, \Phi_2\right) = \frac{di_2}{dx} + ai_0\left(\frac{p\bar{y}_{O_2}}{p^{ref}}\right)\exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\left(-\frac{\alpha_c F}{RT}\right)(\Phi_1 - \Phi_2) + ai_0\left(\frac{p}{p^{ref}}\right)\exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\bar{y}_{O_2} + aC\frac{d(\Phi_1 - \Phi_2)}{dt} \tag{6.36}$$

For Equation 6,

$$f\left(N_{O_2}, y_{O_2}, \Phi_1, \Phi_2\right) = -\frac{p}{RT}\frac{dy_{O_2}}{dt} + \frac{dN_{O_2}}{dx} + \frac{ai_0}{4F}\left(\frac{py_{O_2}}{p^{ref}}\right)\exp\left(-\frac{\alpha_c F}{RT}(\Phi_1 - \Phi_2 - U^0)\right) \tag{6.37}$$

$$f\left(N_{O_2}, y_{O_2}, \Phi_1, \Phi_2\right) = \left.\frac{df}{dN_{O_2}}\right|_{\bar{y}_{O_2},\bar{\Phi}_1,\bar{\Phi}_2} + \left.\frac{df}{dy_{O_2}}\right|_{\bar{N}_{O_2},\bar{\Phi}_1,\bar{\Phi}_2} + \left.\frac{df}{d\Phi_1}\right|_{\bar{N}_{O_2},\bar{y}_{O_2},\bar{\Phi}_2} + \left.\frac{df}{\mathrm{d}\Phi_2}\right|_{\bar{N}_{O_2},\bar{y}_{O_2},\bar{\Phi}_1} \tag{6.38}$$

$$f\left(N_{O_2}, y_{O_2}, \Phi_1, \Phi_2\right) = -\frac{p}{RT}\frac{dy_{O_2}}{dt} + \frac{dN_{O_2}}{dx} + \frac{ai_0}{4F}\left(\frac{p\bar{y}_{O_2}}{p^{ref}}\right)\exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\left(-\frac{\alpha_c F}{RT}\right)(\Phi_1 - \Phi_2) + \frac{ai_0}{4F}\left(\frac{p}{p^{ref}}\right)\exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\bar{y}_{O_2} \tag{6.39}$$

The linearized equations can then be transformed into the frequency domain as described in the first case study. The frequency-domain equations and boundary conditions are listed in Table 6-6.

Table 6-6: Case Study 2 Frequency Domain Equations

$x = 0$			$x = L$
	1	$\tilde{\imath}_1 = -\sigma \frac{d\tilde{\Phi}_1}{dx}$	$\tilde{\Phi}_1 = \Delta\Phi$
	2	$\tilde{\imath}_2 = -\kappa \frac{d\tilde{\Phi}_2}{dx}$	$\tilde{\imath}_2 = 0$
$\tilde{\imath}_1 = 0$	3	$\frac{d\tilde{\imath}_1}{dx} + \frac{d\tilde{\imath}_2}{dx} = 0$	
$\tilde{\Phi}_2 = 0$	4	$\frac{d\tilde{\imath}_2}{dx} = -a i_0 \left(\frac{p\bar{y}_{O_2}}{p^{ref}}\right) \exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\left(-\frac{\alpha_c F}{RT}\right)(\tilde{\Phi}_1 - \tilde{\Phi}_2) - a i_0 \exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\frac{p\tilde{y}_{O_2}}{p^{ref}} + j\omega a C(\tilde{\Phi}_1 - \tilde{\Phi}_2)$	
$\tilde{N}_{O_2} = 0$	5	$\tilde{N}_{O_2} = -D_{O_2,w} \frac{p}{RT} \frac{d\tilde{y}_{O_2}}{dx}$	
	6	$j\omega \frac{p}{RT} \tilde{y}_{O_2} = \frac{d\tilde{N}_{O_2}}{dx} + \frac{a i_0}{4F}\left(\frac{p\bar{y}_{O_2}}{p^{ref}}\right) \exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\left(-\frac{\alpha_c F}{RT}\right)(\tilde{\Phi}_1 - \tilde{\Phi}_2) + \frac{a i_0}{4F} \exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\left(\frac{p\tilde{y}_{O_2}}{p^{ref}}\right)$	$\tilde{y}_{O_2} = 0$

Likewise, the frequency-domain equations can be expanded into their real and imaginary components as described in the first case study. The complete set of equations and boundary conditions split into real and imaginary parts are listed in Table 6-7.

Table 6-7: Case Study 2 Real and Imaginary Frequency Domain Equations

$\boldsymbol{x=0}$			$\boldsymbol{x=L}$
	1	$\tilde{\imath}_{1,Re} = -\sigma \frac{d\tilde{\Phi}_{1,Re}}{dx}$	$\tilde{\Phi}_{1,Re} = \Delta\Phi$
	2	$\tilde{\imath}_{2,Re} = -\kappa \frac{d\tilde{\Phi}_{2,Re}}{dx}$	$\tilde{\imath}_{2,Re} = 0$
$\tilde{\imath}_{1,Re} = 0$	3	$\frac{d\tilde{\imath}_{1,Re}}{dx} + \frac{d\tilde{\imath}_{2,Re}}{dx} = 0$	
$\tilde{\Phi}_{2,Re} = 0$	4	$\frac{d\tilde{\imath}_{2,Re}}{dx} = -ai_0\left(\frac{p\bar{y}_{O_2}}{p^{ref}}\right)\exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\left(-\frac{\alpha_c F}{RT}\right)(\tilde{\Phi}_{1,Re} - \tilde{\Phi}_{2,Re})$ $-ai_0 \exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\frac{p\tilde{y}_{O_2,Re}}{p^{ref}} - \omega a C(\tilde{\Phi}_{1,Im} - \tilde{\Phi}_{2,Im})$	
$\tilde{N}_{O_2,Re} = 0$	5	$\tilde{N}_{O_2,Re} = -D_{O_2,w}\frac{p}{RT}\frac{d\tilde{y}_{O_2,Re}}{dx}$	
	6	$-\omega\frac{p}{RT}\tilde{y}_{O_2,Im} = \frac{d\tilde{N}_{O_2,Re}}{dx} + \frac{ai_0}{4F}\left(\frac{p\bar{y}_{O_2}}{p^{ref}}\right)\exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\left(-\frac{\alpha F}{RT}\right)(\tilde{\Phi}_{1,Re}$ $- \tilde{\Phi}_{2,Re}) + \frac{ai_0}{4F}\exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\left(\frac{p\tilde{y}_{O_2,Re}}{p^{ref}}\right)$	$\tilde{y}_{O_2,Re} = 0$
	7	$\tilde{\imath}_{1,Im} = -\sigma \frac{d\tilde{\Phi}_{1,Im}}{dx}$	$\tilde{\Phi}_{1,Im} = 0$
	8	$\tilde{\imath}_{2,Im} = -\kappa \frac{d\tilde{\Phi}_{2,Im}}{dx}$	$\tilde{\imath}_{2,Im} = 0$
$\tilde{\imath}_{1,Im} = 0$	9	$\frac{d\tilde{\imath}_{1,Im}}{dx} + \frac{d\tilde{\imath}_{2,Im}}{dx} = 0$	
$\tilde{N}_{O_2,Im} = 0$	10	$\frac{d\tilde{\imath}_{2,Im}}{dx} = -ai_0\left(\frac{p\bar{y}_{O_2}}{p^{ref}}\right)\exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\left(-\frac{\alpha_c F}{RT}\right)(\tilde{\Phi}_{1,Im} - \tilde{\Phi}_{2,Im})$ $-ai_0 \exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\frac{p\tilde{y}_{O_2,Im}}{p^{ref}} + \omega a C(\tilde{\Phi}_{1,Re} - \tilde{\Phi}_{2,Re})$	
$\tilde{N}_{O_2,Im} = 0$	11	$\tilde{N}_{O_2,Im} = -D_{O_2,w}\frac{p}{RT}\frac{d\tilde{y}_{O_2,Im}}{dx}$	
	12	$\omega\frac{p}{RT}\tilde{y}_{O_2,Re} = \frac{d\tilde{N}_{O_2,Im}}{dx} + \frac{ai_0}{4F}\left(\frac{p\bar{y}_{O_2}}{p^{ref}}\right)\exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\left(-\frac{\alpha F}{RT}\right)(\tilde{\Phi}_{1,Im}$ $- \tilde{\Phi}_{2,Im}) + \frac{ai_0}{4F}\exp\left(-\frac{\alpha_c F}{RT}(\bar{\Phi}_1 - \bar{\Phi}_2 - U^0)\right)\left(\frac{p\tilde{y}_{O_2,Im}}{p^{ref}}\right)$	$\tilde{y}_{O_2,Im} = 0$

Simulations were carried out using Approach 2 and Approach 3 at several different steady-state voltages. Approach 1 was not included in this case as the magnitude of the perturbation required to maintain linearity throughout the entire impedance spectrum was below the machine error (double precision for MATLAB R2020a). The impedance for this system was calculated using Equation (6.30). The resulting Nyquist plots of the impedance spectra at two different potential values are shown in Figure 6-3. Again, the results for approaches 2 and 3 match exactly in both cases. Approach 2 had a CPU time of 4.5 seconds and Approach 3 had a CPU time of 4.7 seconds.

As the potential decreases, the magnitude of the impedance decreases, and the frequencies increase. Additionally, at around 0.85 V, a small 45° leg develops at the high frequency range of the spectrum. This feature is due to mass transport limitations, which does not occur in the results for case study 1 since the concentration is assumed to be uniform. As the potential continues to decrease, a greater range of the impedance spectra shows diffusion limited behavior.

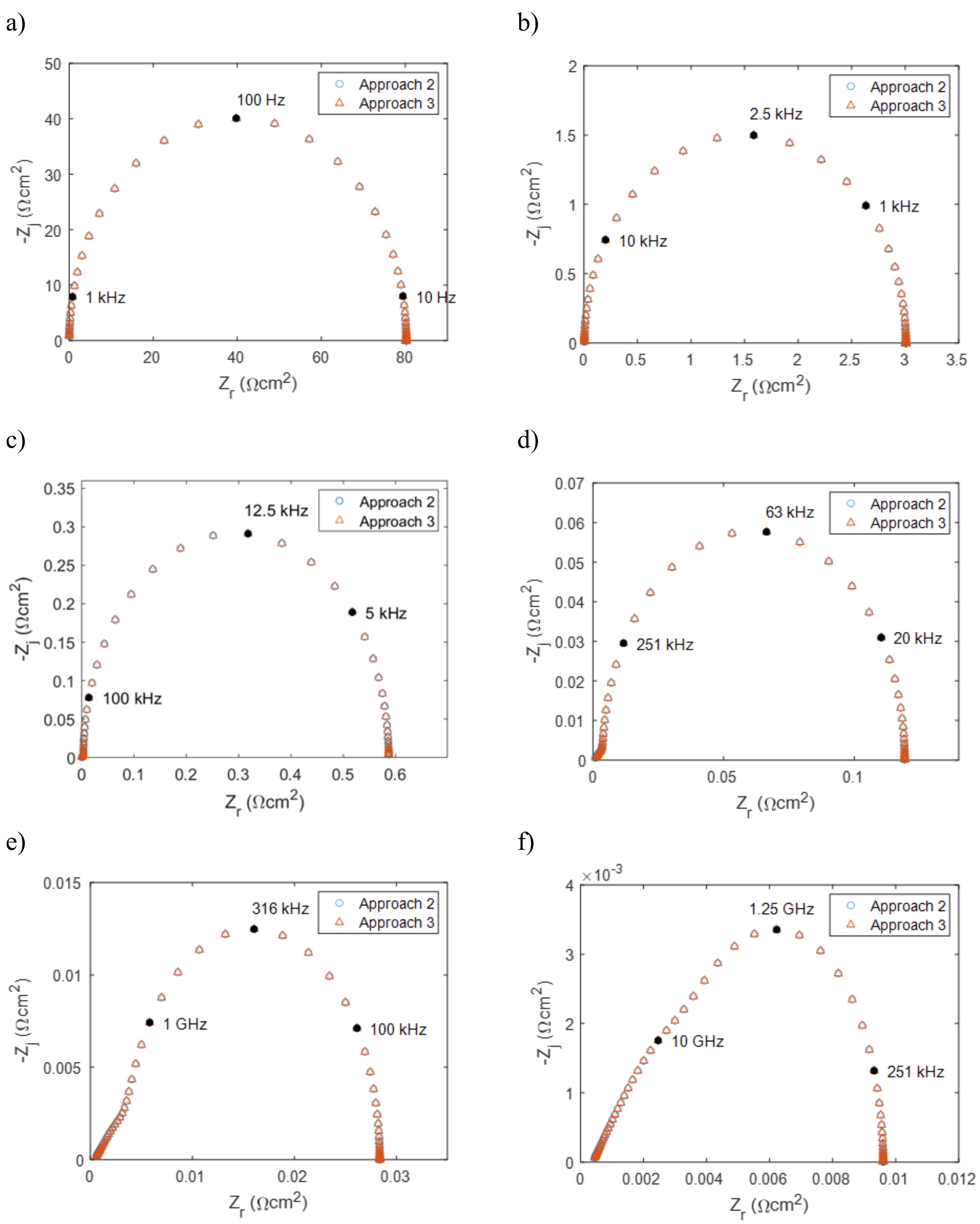
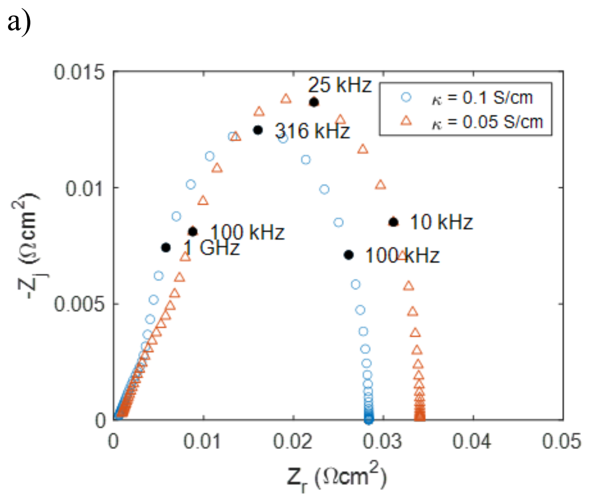


Figure 6-3: Comparison of impedance model simulation techniques for a porous electrode with oxygen reduction reaction with Tafel kinetics. Applied cell potential a) 1.0 V, b) 0.9 V, c) 0.85 V, d) 0.8 V, e) 0.75 V, f) 0.7 V.

Furthermore, this model can be used to carry out sensitivity studies on PEM fuel cell operating conditions and material properties. Typical signs of degradation include loss of membrane conductivity and a decrease in catalyst layer specific interfacial surface area. Results in Figure 6-4 show how impedance spectra at beginning-of-life and end-of-life might differ. A decrease in ionomer conductivity shifts the spectrum to a higher magnitude and lower frequencies. Additionally, the mass-transport limited region of the spectrum increases. A decrease in catalyst layer interfacial surface area shifts the spectrum to a higher magnitude and lower frequencies; however, the diffusion limited region of the spectrum is unchanged.



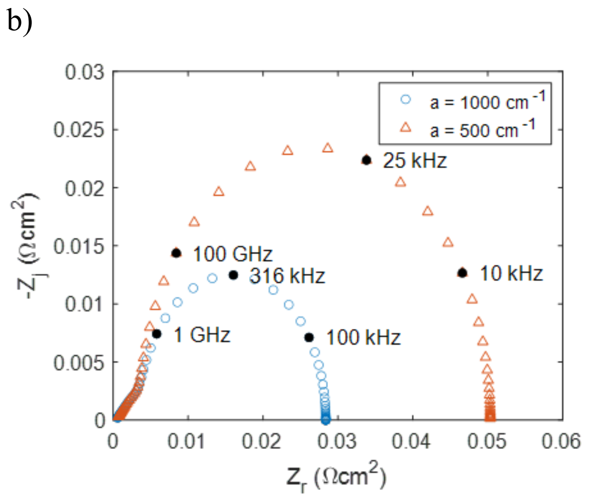


Figure 6-4: Impedance spectra at 0.75 V and different material properties a) ionomer conductivity and b) specific interfacial surface area. Simulations solved using Approach 2.

Chapter 7 – Conclusions and Future Research Directions

With increasing interest in use of PEMFCs in heavy-duty vehicle applications, a deeper understanding of degradation phenomena is needed in order to meet durability targets. Here a modeling approach is used to analyze the interactions between mechanical and chemical degradation of fuel cell membranes. A fully coupled one-dimensional, non-isothermal, single-phase, transient fuel-cell performance and membrane mechanical model have been developed and directly coupled. Simulation results demonstrated that the model predicts fuel-cell performance and growth of pinholes present in the membrane by accounting for the various physics and both direct and indirect interactions. The model results demonstrate the importance of coupling the transport model with the mechanical model, as well as the addition of the chemical degradation kinetics. The change in mechanical properties as a result of membrane degradation by radical attack accelerates mechanical defect growth, which then leads to additional gas crossover and drives the degradation cycle.

The fuel-cell degradation model with coupled transport, mechanical degradation, and chemical degradation can be improved to further capture the synergistic relationships with degradation phenomena. Currently, the mechanical model includes the assumption that pinhole deformation only occurs under plastic strain. The accuracy of the model can be improved by including pinhole closure and viscoelastic effects. Additionally, incorporating initiation conditions for a defect to occur can further demonstrate the synergistic effects of mechanical and chemical degradation. Furthermore, the effectiveness of mitigation approaches can be evaluated by expanding the model to include mechanical reinforcement into the PEM to improve chemical-mechanical stability.

The coupled full-cell, transient fuel-cell performance model was improved upon with the addition of a microkinetic framework for degradation and concentrated-solution-theory based transport and mitigating effects of cerium ion. The model predicted the migration of cerium out of the membrane into the catalyst layers, with the cerium primarily accumulating in the cathode. Simulation results agree with a decrease in FRR and OCV drop as the cerium concentration increases. A comparison of results between dilute-solution-theory and concentrated-solution-theory demonstrate that the dilute-solution-theory overestimates the migration force acting on cerium, which leads to an accumulation of cerium ions in the cathode catalyst layer and subsequent steep drop in limiting current densities at high cerium content. A voltage-loss breakdown shows that cerium leads to voltage losses in the cell due to both proton activity loss and modification of membrane transport properties, and these losses occur simultaneously and are comparable in magnitude. These losses scale exponentially with current density until the limiting current density is reached. The concentrated-solution-theory model corrects for this effect by accounting for the interaction between cerium ions and water in the membrane.

Transient simulation results show that the majority of the benefits to chemical degradation mitigation can be achieved at <1% cerium content in the membrane (with the assumed 1-D, single-phase model), at which point the decrease in performance is largely outweighed by the degradation mitigation increase. Additional analysis shows that the time to failure is roughly linear with cerium content at low cerium content, where the slope is dependent on the membrane and catalyst-layer thicknesses. While optimizing performance and durability, thicker membranes and thinner catalyst layers should be considered commensurate with design limitations. Extensions to the model include incorporation of metal ions and radical generation via Fenton's reaction and explicit consideration of cerium ions in the 4+ charge state in the concentrated-solution-theory equations, as well as the addition of multiphase phenomena for modeling higher relative-humidity conditions. The model could also be modified to include higher dimensional effects such as along-the-channel or land/channel distribution of cerium.

A multiphase model was used to analyze the effects of high humidity on membrane degradation. Under high humidity conditions, some of the water vapor condenses to form liquid water. When defect such as pinhole is present in the membrane, the condensed water will prevent gas crossover through the pinhole. Additionally, liquid water also reduces the rate of hydrogen peroxide generation that causes chemical degradation by reducing the amount of catalyst surface area available for reaction. However, this also leads to a decrease in fuel cell performance.

To analyze the effects of cerium mitigation with coupled mechanical and chemical durability, the mechanical model for a pinhole in the membrane was added to the model with microkinetic chemical degradation and cerium transport. The model results show that the presence of a pinhole in the membrane improves hydration throughout the cell and leads to the distribution of cerium to drive towards the anode and membrane, counteracting some of the migration force driving cerium to accumulate in the cathode. Additionally, for the range of pinhole sizes of radius $R_0 \leq 500$ µm, the FRR decreases due to the improved hydration and more even distribution of cerium throughout the cell. Under relative-humidity cycling conditions, the model exhibits an increased pinhole growth rate with higher cerium content. However, these results have not taken into account the effect of cerium on the membrane modulus and other mechanical properties; further analysis is needed. Voltage-cycling conditions demonstrate a decrease in FRR over time as well as an increase in gas crossover at higher cerium contents. Directions for future work include incorporating mechanical properties as a function of cerium content, similar to how mechanical properties are dependent upon the FRR. Additionally, the model does not take into account the effect of chemical degradation on pinhole growth. Localized chemical degradation at the pinhole could cause the pinhole to grow rapidly and lead to accelerated degradation rates.

Electrochemical impedance spectroscopy is a diagnostic tool that is frequently used to characterize PEMFC performance and degradation. Three numerical modeling approaches are described for the simulation of electrochemical impedance response for porous electrodes. The methods presented for building an electrochemical impedance model can be applied to build a physics-based PEMFC impedance model. While a transient-based impedance-model approach could in theory work for the PEMFC degradation model presented in this work, the limitations of this approach, including computation time and accuracy needed for calculation of the impedance response, prevent it. Another approach to developing physics-based impedance models for PEMFCs is to transform the transient model into the frequency domain and linearizing about the steady-state value. In the case studies presented, this approach outperformed the transient-based approach both in accuracy of results and computational time. A simple model for a cathode catalyst layer in a PEMFC with Tafel kinetics for ORR and oxygen diffusion can simulate some of the primary features of the overall cell impedance response. Additionally, this model can be used to carry out sensitivity studies on operating conditions and material properties that may change as a result of membrane degradation. However, the high degree of coupling and nonlinear equations in the full cell model makes the derivation of the impedance model equations impractical. A modified BAND(J) algorithm could be developed to linearize numerically the transient system of equations and transform the model to the frequency domain.

Nomenclature

Roman

a_i	activity of species i
a	electrode specific interfacial area (1/cm)
A	fuel cell area (cm^2)
c	circular pinhole plastic deformation constant
c_i	concentration of species i (mol/cm^3)
c_T	total gas concentration (mol/cm^3)
$\hat{C}_p$	average heat capacity (J/mol·K)
$\mathcal{D}_{ij}$	binary diffusion coefficient for species i and j (cm^2/s)
$\mathcal{D}_{K_i}$	Knudsen diffusion coefficient for species i (cm^2/s)
E	Young's modulus (MPa)
E_h	effectiveness factor for reaction h
E_A	activation energy (J/mol)
EW	equivalent weight (g/equiv)
f	volume fraction of water in the membrane
f_{Ce}	fraction of SO_3^- sites in the membrane that are occupied by cerium ions
F	Faraday's constant (96485 C/equiv)
G	geometric factor
h	material hardening parameter
$i_{0,h}$	Exchange current density for reaction h (A/cm^2)
i_k	superficial current density in phase k (A/cm^2)
$\mathbf{i}$	current density (A/cm^2)
I	ionic strength
j	square root of -1
k	absolute permeability (cm^2)
k	thermal conductivity (W/cm K)
k	inverse Debye length (m^{-1})
k'_h	Thiele modulus reaction rate for reaction h (mol/bar·cm^3·s)
K_{ij}	friction coefficient between species i and j
$\mathcal{K}_i$	hydrodynamic friction coefficient of species i
L_n	thickness of layer n (cm)
m	scaling exponent for Young's modulus
n_h	number of electrons in electrochemical reaction h
n_i	total number of moles of species i
$\mathbf{N}_\mathrm{i}$	flux of species i (mol/cm^2/s)
M_i	molecular weight of species i (g/mol)
p	scaling exponent for yield strength

p_c	critical pressure (bar)
p_g	gas pressure (bar)
p_i	partial pressure of species i (bar)
p_i^{ref}	reference pressure for species i (bar)
p_i^{vap}	vapor pressure for species i (bar)
p_L	liquid pressure (bar)
$p_{L,M}$	Liquid pressure in the membrane (bar)
p	total pressure (bar)
r	radius (cm)
r_c	critical radius of pore hype h (μm)
r_K	Knudsen radius (μm)
R	ideal gas constant (8.314 J/mol·K)
R_i	reaction rate of species i (mol/cm^2/s)
R_{pore}	pore radius (nm)
S	saturation
S_G^0	residual gas saturation
S_L^0	residual liquid saturation
t	time (s)
t_i	transference number of species i
T	temperature (K)
T_{ref}	reference temperature (303.15 K)
U_0^{h}	equilibrium potential of reaction h (V)
u_i	ionic mobility of species i (cm^2/V/s)
$\mathbf{v}$	superficial velocity (cm/s)
$\bar{V}_i$	partial molar volume of species i (cm^3/mol)
$\tilde{V}_i$	effective molar viscous volume of species i
w_i	mass fraction of species i
$\bar{x}$	steady-state value of variable x
$\tilde{x}$	oscillating value of variable x
y_i	mole fraction of species i
z_i	valence of species i
$Z(\omega)$	impedance at frequency ω (Ωcm^2)

Greek

α	membrane transport coefficient (mol^2/J·cm·s)
α_a	anode transfer coefficient
α_c	cathode transfer coefficient
α_{ij}	transport coefficient for species i and j

β	ratio of the effective pore radius and the true pore radius
γ	surface tension (N/m)
ε_0	volume fraction for gas transport
ε_0	vacuum permittivity (8.85×10^{-12} F/m)
ε_M	volume fraction of membrane
ε_r	bulk solvent dielectric constant
$\bar{\varepsilon}$	equivalent strain
η	viscosity (Pa·s)
η_h	overpotential of reaction h (V)
θ_c	critical angle (degrees)
θ_i	distribution factor for species i in a hydrophilic domain
ϑ_h	a function that equals 1 for hydrophilic pores and -1 for hydrophobic pores
κ	conductivity (S/cm)
λ	water content
μ	viscosity (bar·s)
μ_i	chemical potential of species i (J/mol)
ν	Poisson's ratio
ν_i	Stoichiometric coefficient of species i
ξ	electro-osmotic coefficient
Π_h	Peltier coefficient of reaction h (V)
ρ	molar density (mol/cm^3)
ϱ	electrostatic parameter
σ	bulk-phase conductivity (S/cm)
$\bar{\sigma}_e$	equivalent stress (MPa)
σ_i	stress in the i-direction (MPa)
σ_m	mean stress
σ_Y	yield strength (MPa)
τ	tortuosity
τ	time constant (s)
τ_M	tortuosity for membrane with cerium
ϕ	Thiele modulus
ϕ_{mt}	Thiele modulus mass transport (bar·cm·s/mol)
Φ	potential (V)
ψ_i	permeability of species i (mol/bar/cm/s)
ω	angular frequency (rad/s)

Superscripts and Subscripts

∞	infinite dilution

0	initial value
1	electronically conducting phase
2	proton conducting phase
a	anode
c	cathode
Ce	cerium
dry	dry polymer
eff	effective
el	elastic
H	protons
HI	hydrophilic
HO	hydrophobic
hole	pinhole in the membrane
in	inlet through the gas channel
L	liquid
mem	membrane
pl	plastic
sw	swelling
V	vapor
w	water
x	in the x-direction
y	in the y-direction
z	in the z-direction

Abbreviations

CL	catalyst layer
FRR	fluoride release rate
GC	gas channel
GDL	gas diffusion layer
HOR	hydrogen oxidation reaction
Mem	Membrane
OCV	open circuit voltage
ORR	oxygen reduction reaction
PEM	proton-exchange membrane
PEMFC	proton-exchange-membrane fuel cell
PFSA	perfluorosulfonic acid
RH	relative humidity
SHE	standard hydrogen electrode

References

1. Fuel Cell Technologies Program Multi-Year Research, Development and Demonstration Plan, in, F. C. T. Office Editor, U.S. Department of Energy (2017).
2. C. S. Gittleman, F. D. Coms and Y.-H. Lai, in *Polymer Electrolyte Fuel Cell Degradation*, M. M. Mench, E. C. Kumbur and T. N. Veziroglu Editors, p. 15, Academic Press, Waltham, MA (2012).
3. A. Kusoglu and A. Z. Weber, *Journal of Physical Chemistry Letters*, **6**, 4547 (2015).
4. S. Kreitmeier, P. Lerch, A. Wokaun and F. N. Buchi, *Journal of The Electrochemical Society*, **160**, F456 (2013).
5. R. Mukundan, A. M. Baker, A. Kusoglu, P. Beattie, S. Knights, A. Z. Weber and R. L. Borup, *Journal of The Electrochemical Society*, **165**, F3085 (2018).
6. Y. H. Lai, K. M. Rahmoeller, J. H. Hurst, R. S. Kukreja, M. Atwan, A. J. Maslyn and C. S. Gittleman, *Journal of the Electrochemical Society*, **165**, F3217 (2018).
7. A. Kusoglu, A. M. Karlsson, M. H. Santare, S. Cleghorn and W. B. Johnson, *Journal of Power Sources*, **161**, 987 (2006).
8. A. Kusoglu, A. M. Karlsson, M. H. Santare, S. Cleghorn and W. B. Johnson, *Journal of Power Sources*, **170**, 345 (2007).
9. T. Uchiyama, M. Kato, Y. Ikogi and T. Yoshida, *Journal of Fuel Cell Science and Technology*, **9** (2012).
10. A. Kusoglu, M. H. Santare and A. M. Karlsson, *Journal of Polymer Science Part B: Polymer Physics*, **49**, 1506 (2011).
11. G. Ding, M. H. Santare, A. M. Karlsson and A. Kusoglu, *Journal of Power Sources*, **216**, 114 (2016).
12. A. Kusoglu and A. Z. Weber, *Journal of The Electrochemical Society*, **161**, E3311 (2014).
13. A. Z. Weber, *Journal of The Electrochemical Society*, **155**, B521 (2008).
14. M. Zaton, J. Roziere and D. J. Jones, *Sustainable Energy & Fuels*, **1**, 409 (2017).
15. F. D. Coms, *ECS Transactions*, **16**, 235 (2008).
16. T. Kinumoto, M. Inaba, Y. Nakayama, K. Ogata, R. Umebayashi, A. Tasaka, Y. Iriyama, T. Abe and Z. Ogumi, *Journal of Power Sources*, **158**, 1222 (2006).
17. A. Pozio, R. F. Silva, M. D. Francesco and L. Giorgi, *Electrochimica Acta*, **48**, 1543 (2003).
18. R. Lin, E. Gulzow, M. Schulze and K. A. Friedrich, *Journal of The Electrochemical Society*, **158**, B11 (2011).
19. D. Liu and S. Case, *Journal of Power Sources*, **162**, 521 (2006).
20. L. Dubau, L. Castanheira, F. Maillard, M. Chatenet, O. Lottin, G. Maranzana, J. Dillet, A. Lamibrac, J.-C. Perrin, E. Moukheiber, A. ElKaddouri, G. D. Moor, C. Bas, L. Flandin and N. Caque, *Wiley Interdisciplinary Reviews: Energy and Environment*, **3**, 540 (2014).
21. R. Borup, J. Meyers, B. Pivovar, Y. S. Kim, R. Mukundan, N. Garland, D. Myers, M. Wilson, F. Garzon, D. Wood, P. Zelenay, K. More, K. Stroh, T. Zawodzinski, J. Boncella, J. E. McGrath, M. Inaba, K. Miyatake, M. Hori, K. Ota, Z. Ogumi, S. Miyata, A. Nishikata, Z. Siroma, Y. Uchimoto, K. Yasuda, K.-i. Kimijima and N. Iwashita, *Chemical Reviews*, **107**, 3904 (2007).
22. F. D. Coms, H. Liu and J. E. Owejan, *ECS Transactions*, **16**, 1735 (2008).
23. P. Trogadas, J. Parrondo and V. Ramani, *Electrochem Solid St*, **11**, B113 (2008).
24. P. Trogadas, J. Parrondo and V. Ramani, *ACS Appl Mater Inter*, **4**, 5098 (2012).

25. S. M. Stewart, D. Spernjak, R. Borup, A. Datye and F. Garzon, *ECS Electrochem Lett*, **3**, F19 (2014).
26. F. D. Coms, S. Schlick and M. Danilczuk, in *The Chemistry of Membranes Used in Fuel Cells: Degradation and Stabilization*, 1st ed., S. Schlick Editor, John Wiley & Sons, Inc. (2018).
27. M. Zaton, B. Prelot, N. Donzel, J. Roziere and D. J. Jones, *J Electrochem Soc*, **165**, F3281 (2018).
28. A. Z. Weber and C. Delacourt, *Fuel Cells*, **0**, 459 (2008).
29. J. Xie, S. Ban, B. Liu and H. Zhou, *Applied Surface Science*, **362**, 441 (2016).
30. M. A. Quiroga, K. Malek and A. A. Franco, *Journal of The Electrochemical Society*, **163**, F59 (2016).
31. T. Jahnke, G. Futter, A. Latz, T. Malkow, G. Papakonstantinou, G. Tsotridis, P. Schott, M. Gerard, M. Quinaud, M. Quiroga, A. A. Franco, K. Malek, F. Calle-Vallejo, R. F. d. Morais, T. Kerber, P. Sautet, D. Loffreda, S. Strahl, M. Serra, P. Polvreino, C. Pianese, M. Mayur, W. G. Bessler and C. Kompis, *Journal of Power Sources*, **304**, 207 (2016).
32. R. Solasi, Y. Zou, X. Huang, K. Reifsnider and D. Condit, *Journal of Power Sources*, **167**, 366 (2007).
33. M. Hasan, A. Goshtasbi, J. Chen, M. H. Santare and T. Ersal, *Journal of The Electrochemical Society*, **165**, F3359 (2018).
34. L. Gubler, S. M. Dockheer and W. H. Koppenol, *Journal of The Electrochemical Society*, **158**, B755 (2011).
35. L. Gubler and W. H. Koppenol, *Journal of The Electrochemical Society*, **159**, B211 (2012).
36. K. H. Wong and E. Kjeang, *Journal of The Electrochemical Society*, **161**, F823 (2014).
37. K. H. Wong and E. Kjeang, *ChemSusChem*, **8**, 1072 (2015).
38. K. H. Wong and E. Kjeang, *Journal of The Electrochemical Society*, **164**, F1179 (2017).
39. K. H. Wong and E. Kjeang, *Journal of The Electrochemical Society*, **166**, F128 (2019).
40. R. Singh, P. C. Sui, K. H. Wong, E. Kjeang, S. Knights and N. Djilali, *Journal of The Electrochemical Society*, **165**, F3328 (2018).
41. G. A. Futter, A. Latz and T. Jahnke, *Journal of Power Sources*, **410-411**, 78 (2019).
42. T. Fuller, Solid-polymer-electrolyte fuel cells, in *Chemical Engineering*, University of California Berkeley (1992).
43. T. F. Fuller and J. Newman, *Journal of The Electrochemical Society*, **139**, 1332 (1992).
44. A. Z. Weber and J. Newman, *Journal of The Electrochemical Society*, **150**, A1008 (2003).
45. A. Z. Weber and J. Newman, *Journal of The Electrochemical Society*, **151**, A311 (2004).
46. A. Z. Weber and J. Newman, *Journal of The Electrochemical Society*, **151**, A326 (2004).
47. A. Z. Weber, Modeling Water Management in Polymer-Electrolyte Fuel Cells, in *Chemical Engineering*, University of California, Berkeley (2004).
48. A. Z. Weber and J. Newman, *Journal of The Electrochemical Society*, **153**, A2205 (2006).
49. I. V. Zenyuk, P. K. Das and A. Z. Weber, *Journal of The Electrochemical Society*, **163**, F691 (2016).
50. F. A. L. Dullien and H. Brenner, *Porous Media: Fluid Transport and Pore Structure*, Elsevier Science (2012).
51. A. Kusoglu and A. Z. Weber, *Chemical Reviews*, **117**, 987 (2017).
52. J. Newman, *Industrial & Engineering Chemistry Fundamentals*, **7**, 514 (1968).
53. J. Newman and K. K. Thomas-Alyea, *Electrochemical Systems*, John Wiley & Sons, Inc. (2004).

54. S. Kreitmeier, M. Michiardi, A. Wokaun and F. N. Buchi, *Electrochimica Acta*, **80**, 240 (2012).
55. S. Kreitmeier, G. A. Schuler, A. Wokaun and F. N. Buchi, *Journal of Power Sources*, **212**, 139 (2012).
56. Y. Wang and C.-Y. Wang, *Journal of Power Sources*, **147**, 148 (2005).
57. S. W. Shi, A. Z. Weber and A. Kusoglu, *Journal of Membrane Science*, **516**, 123 (2016).
58. A. Kusoglu, Y. Tang, M. Lugo, A. M. Karlsson, M. H. Santare, S. Cleghorn and W. B. Johnson, *Journal of Power Sources*, **195**, 483 (2010).
59. A. Kusoglu, A. M. Karlsson and M. H. Santare, *Polymer*, **51**, 1457 (2010).
60. A. Kusoglu, M. H. Santare, A. M. Karlsson, S. Cleghorn and W. B. Johnson, *Journal of The Electrochemical Society*, **157**, B705 (2010).
61. M. Inaba, T. Kinumoto, M. Kiriake, R. Umebayashi, A. Tasaka and Z. Ogumi, *Electrochimica Acta*, **51**, 5746 (2006).
62. K. Teranishi, K. Kawata, S. Tsushima and S. Hirai, *Electrochemical and Solid-State Letters*, **9**, A475 (2006).
63. S. Kundu, M. W. Fowler, L. C. Simon, R. Abouatallah and N. Beydokhti, *Journal of Power Sources*, **183**, 619 (2008).
64. E. K. Unnikrishnan, S. D. Kumar and B. Maiti, *Journal of Membrane Science*, **137**, 133 (1997).
65. S. Kundu, L. C. Simon and M. W. Fowler, *Polymer Degradation and Stability*, **93**, 214 (2008).
66. A. P. Young, J. Stumper, S. Knights and E. Gyenge, *Journal of The Electrochemical Society*, **157**, B425 (2010).
67. A. Kusoglu, M. Calabrese and A. Z. Weber, *ECS Electrochemistry Letters*, **3**, F33 (2014).
68. J. Newman and K. E. Thomas-Alyea, *Electrochemical Systems*, John Wiley & Sons, Inc., Hoboken, New Jersey (2004).
69. Y. Wang and C.-Y. Wang, *Electrochimica Acta*, **50**, 1307 (2005).
70. F. A. d. Bruijn, V. A. T. Dam and G. J. M. Janssen, *Fuel Cells*, **8**, 3 (2008).
71. C. S. Gittleman, F. D. Coms and Y.-H. Lai, in *Polymer Electrolyte Fuel Cell Degradation*, M. M. Mench, E. C. Kumbur and T. N. Veziroglu Editors, p. 15, Academic Press, Boston (2012).
72. L. M. Pant, M. R. Gerhardt, N. Macauley, R. Mukundan, R. L. Borup and A. Z. Weber, *Electrochimica Acta*, **326**, 134963 (2019).
73. A. R. Crothers, R. M. Darling, A. Kusoglu, C. J. Radke and A. Z. Weber, *Journal of The Electrochemical Society*, **167**, 013548 (2020).
74. A. R. Crothers, R. M. Darling, D. I. Kushner, M. L. Perry and A. Z. Weber, *Journal of The Electrochemical Society*, **167**, 013549 (2020).
75. A. Kusoglu and A. Z. Weber, *ECS Transactions*, **58**, 999 (2013).
76. S. v. Venkatesan, C. Lim, S. Holdcroft and E. Kjeang, *Journal of The Electrochemical Society*, **163**, F637 (2016).
77. A. M. Baker, A. R. Crothers, K. Chintam, X. Luo, A. Z. Weber, R. L. Borup and A. Kusoglu, *ACS Applied Polymer Materials* (2020).
78. A. M. Baker, R. Mukundan, D. Spernjak, E. J. Judge, S. G. Advani, A. K. Prasad and R. L. Borup, *Journal of The Electrochemical Society*, **163**, F1023 (2016).
79. C. W. Monroe and J. Newman, *Industrial & Engineering Chemistry Research*, **45**, 5361 (2006).

80. P. N. Pintauro and D. N. Bennion, *Industrial & Engineering Chemistry Fundamentals*, **23**, 230 (1984).
81. C. Delacourt and J. Newman, *Journal of The Electrochemical Society*, **155**, B1210 (2008).
82. *CRC Handbook of Chemistry and Physics*, CRC Press, Cleveland, Ohio (1978).
83. N. Ouerfelli and M. Bouanz, *Journal of Physics: Condensed Matter*, **8**, 2763 (1996).
84. T. W. Chapman, The Transport Properties of Concentrated Electrolyte Solutions, in, University of California, Berkeley (1967).
85. A. M. Baker, S. K. Babu, R. Mukudan, S. G. Advani, A. K. Prasad, D. Spernjak and R. L. Borup, *Journal of The Electrochemical Society*, **164**, F1272 (2017).
86. V. M. Ehlinger, A. Kusoglu and A. Z. Weber, *Journal of The Electrochemical Society*, **166**, F3255 (2019).
87. V. M. Ehlinger, A. R. Crothers, A. Kusoglu and A. Z. Weber, *Journal of Physics: Energy* (2020).
88. M. E. Orazem and B. Tribollet, *Electrochemical Impedance Spectroscopy*, John Wiley & Sons, Inc., Hoboken, NJ (2017).
89. S. K. Roy, M. E. Orazem and B. Tribollet, *Journal of The Electrochemical Society*, **154**, B1378 (2007).
90. M. Heinzmann, A. Weber and E. Ivers-Tiffée, *Journal of Power Sources*, **444**, 227279 (2019).
91. X. Yuan, H. Wang, J. C. Sun and J. Zhang, *International Journal of Hydrogen Energy*, **32**, 4365 (2007).
92. B. Tribollet and J. Newman, *Journal of The Electrochemical Society*, **131**, 2780 (1984).
93. R. Pollard and T. Comte, *Journal of The Electrochemical Society*, **136**, 3734 (1989).
94. M. Doyle, J. P. Meyers and J. Newman, *Journal of The Electrochemical Society*, **147**, 99 (2000).
95. J. P. Meyers, M. Doyle, R. M. Darling and J. Newman, *Journal of The Electrochemical Society*, **147**, 2930 (2000).
96. J. R. Vang, S. J. Andreasen and S. K. Kaer, *Journal of Fuel Cell Science and Technology*, **9**, 021005 (2012).
97. A. Kosakian, L. P. Urbina, A. Heaman and M. Secanell, *Electrochimica Acta*, **350**, 136204 (2020).
98. Q. Guo and R. E. White, *Journal of The Electrochemical Society*, **151**, E133 (2004).
99. Q. Guo, M. Cayetano, Y.-m. Tsou, E. S. D. Castro and R. E. White, *Journal of The Electrochemical Society*, **150**, A1440 (2003).
100. P. M. Gomadam and J. W. Weidner, *International Journal of Energy Research*, **29**, 1133 (2005).
101. A. A. Kulikovsky, *Journal of Electroanalytical Chemistry*, **669**, 28 (2012).
102. A. A. Kulikovsky, *Electrochimica Acta*, **147**, 773 (2014).
103. A. A. Kulikovsky, *Journal of The Electrochemical Society*, **162**, F217 (2015).
104. A. A. Kulikovsky, *Electrochimica Acta*, **247**, 730 (2017).
105. A. Kulikovsky, *eTransportation*, **100026** (2019).
106. A. Kulikovsky, *Journal of The Electrochemical Society*, **166**, F306 (2019).
107. J. S. Newman and C. W. Tobias, *Journal of The Electrochemical Society*, **109**, 1183 (1962).
108. A. A. Kulikovsky, *Electrochimica Acta*, **55**, 6391 (2010).

Appendix A – BANDmaps/list of equations and boundary conditions

A full list of all variables, boundary conditions, and steady-state equations used in the model is shown. The equations are divided into the five modeling domains: anode gas diffusion layer, anode catalyst layer, membrane, cathode catalyst layer, and cathode gas diffusion layer. The boundary conditions as well as the gas channel/gas diffusion layer boundaries are listed in between each layer.

Variable	Symbol
current density in the solid phase	i_1
potential in the solid phase	Φ_1
current density in the membrane phase	i_2
potential in the membrane phase	Φ_2
water chemical potential	μ_w
flux of water in the membrane phase	$N_{w,M}$
oxygen flux	N_{O_2}
nitrogen flux	N_{N_2}
water vapor flux	N_w
hydrogen flux	N_{H_2}
oxygen mole fraction	y_{O_2}
nitrogen mole fraction	y_{N_2}
water vapor mole fraction	y_w
hydrogen mole fraction	y_{H_2}
temperature	T
pressure	p
membrane thickness	ℓ
membrane expansion fraction	τ
liquid pressure	p_L
liquid pressure in the membrane	$p_{L,M}$
liquid water flux	$N_{w,L}$
cerium doping fraction	f_{Ce}
cerium chemical potential	μ_{Ce}
cerium flux	N_{Ce}
total moles of cerium	n_{Ce}
flux of hydrogen peroxide	$N_{H_2O_2}$
concentration of hydrogen peroxide	$c_{H_2O_2}$

concentration of sulfonic acid sites in membrane	$c_{R_fSO_3}$
flux of hydrogen fluoride	N_{HF}
concentration of hydrogen fluoride	c_{HF}
concentration of hydroxyl ions	$c_{OH^\bullet}$
concentration of first degradation reaction of sulfonic acid sites	$c_{R_f\alpha O^\bullet}$
concentration of second degradation reaction of sulfonic acid sites	$c_{R_f\beta O^\bullet}$
concentration of end-chain sites	c_{R_fCOOH}

Equations

$$i_2 = \left(-\frac{\kappa t_{\mathrm{H}^+}}{F}\nabla\Phi_2 - \left(\alpha_{H^+w} + \frac{\kappa\xi t_{H^+}}{F^2}\right)\nabla\mu_w - \left(\alpha_{H^+Ce} + \frac{t_{H^+}t_{Ce}\kappa}{z_{Ce}F^2}\right)\nabla\mu_{Ce} + z_{Ce}N_{Ce}\right)F \quad \text{(A.1)}$$

$$N_{w,M} = -\left(\alpha_{Cew} + \frac{\xi t_{Ce}\kappa}{z_{Ce}F^2}\right)\nabla\mu_{Ce} - \frac{\kappa\xi}{F}\nabla\Phi_2 - \left(\alpha_{ww} + \frac{\kappa\xi^2}{F^2}\right)\nabla\mu_w \quad \text{(A.2)}$$

$$\nabla\cdot N_w = k_{M,V}\left(\mu_w - \bar{V}_w p - RT\log\left(\frac{y_w p}{P_w^{sat}}\right)\right) \quad \text{(A.3)}$$

$$\rho\hat{C}_p\left(\frac{\partial T}{\partial t} + \mathbf{v}\cdot\nabla T\right) = \nabla\cdot(k\nabla T) + \frac{\mathbf{i}\cdot\mathbf{i}}{\kappa} + \sum_h i_h(\eta_h + \Pi_\mathrm{h}) \quad \text{(A.4)}$$

$$\nabla\mu_{Ce} = \frac{RT}{c_{Ce}}\nabla c_{Ce} - \frac{z_{Ce}}{z_H}\frac{RT}{c_H}\nabla c_H \quad \text{(A.5)}$$

$$N_{Ce} = -\left(\alpha_{CeCe} + \left(\frac{t_{Ce}}{z_{Ce}}\right)^2\frac{\kappa}{F^2}\right)\nabla\mu_{Ce} - \left(\alpha_{Cew} + \frac{\xi t_{Ce}\kappa}{z_{Ce}F^2}\right)\nabla\mu_w - \frac{t_{Ce}\kappa}{z_{Ce}F}\nabla\Phi_2 \quad \text{(A.6)}$$

$$\nabla\cdot i_2 = i_{HOR} + i_{ORR_{4e^-}} + i_{ORR_{2e^-}} \quad \text{(A.7)}$$

$$\nabla\cdot N_{w,m} = \frac{1}{2F}i_{ORR_{4e^-}} - k_{M,V}\left(\mu_w - \bar{V}_w p - RT\log\left(\frac{y_w p}{P_w^{sat}}\right)\right) \quad \text{(A.8)}$$

$$\nabla\cdot N_{O_2} = -\frac{1}{4F}i_{ORR_{4e^-}} - \frac{1}{2F}i_{ORR_{2e^-}} \quad \text{(A.9)}$$

$$i_2 = -\left(\kappa_L\nabla\Phi_2 + \frac{\kappa_L\xi_L}{F}\left(\bar{V}_w\nabla p_{L,M}\right)\right)S - \left(\kappa_v\nabla\Phi_2 + \frac{\kappa_V\xi_V}{F}\nabla\mu_w\right)(1-S) \quad \text{(A.10)}$$

$$p_{L,M} = N_{w,M} - \left(\frac{\kappa_L \xi_L}{F} \nabla \Phi_2 + \left(\alpha_L + \frac{\kappa_L \xi_L^2}{F^2}\right)\left(\bar{V}_w \nabla p_{L,mem}\right)\right) S$$

$$- \left(\frac{\kappa_V \xi_V}{F} \nabla \Phi_2 + \left(\alpha_V + \frac{\kappa_V \xi_V^2}{F^2}\right) \nabla \mu_w\right)(1 - S) \quad \text{(A.11)}$$

$$\nabla \cdot N_{w,m} = \frac{1}{2F} i_{ORR_{4e^-}} - k_{M,V}\left(\mu_w - \bar{V}_w p - RT \log\left(\frac{y_w p}{P_w^{sat}}\right)\right) - k_{M,L}\left(p_{L,M} - p_L\right) \quad \text{(A.12)}$$

$$\nabla \cdot N_w = k_{M,V}\left(\mu_w - \bar{V}_w p - RT \log\left(\frac{y_w p}{P_w^{sat}}\right)\right) + k_{M,L}\left(p_{L,M} - p_L\right)$$

$$- \frac{k_{evap}\left(p_g y_w - p_w^{sat}\right)}{RT} \quad \text{(A.13)}$$

B.1 Mechanical Degradation Model with Empirical Chemical Degradation

Anode Diffusion Media

Variable	**Anode Gas Channel / Anode Diffusion Media Boundary**	**Anode Diffusion Media**	**Anode Diffusion Media / Anode Catalyst Layer Boundary**
i_1	$\nabla \cdot i_1 + \nabla \cdot i_2 = 0$		
Φ_1	$\Phi_1 = 0$	$i_1 = -\sigma \nabla \Phi_1$	
i_2	$i_2 = 0$		
Φ_2	$\Phi_2 = 0$		$i_2 = -\kappa \nabla \Phi_2 - \frac{\kappa \xi}{F} \nabla \mu_w$
μ_w	$\mu_w = 0$		$N_{w,M} = \frac{\kappa \xi}{F} \nabla \Phi_2 + \left(\alpha + \frac{\kappa \xi^2}{F^2} \right) \nabla \mu_w$
$N_{w,M}$	$N_{w,M} = 0$		
N_{O_2}	$N_{O_2} = 0$	$\nabla \cdot N_{O_2} = 0$	Equation A.9
N_{N_2}	Gas Channel Mass Balance	$\nabla \cdot N_{N_2} = 0$	
N_w	Gas Channel Mass Balance	$\nabla \cdot N_w = 0$	Equation A.3
N_{H_2}	$\sum_i y_i = 1$	$\nabla \cdot N_{H_2} = 0$	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$
y_{O_2}	Stefan-Maxwell		
y_{N_2}	$y_{N_2} = 0$	Stefan-Maxwell	
y_w	Stefan-Maxwell		
y_{H_2}	Stefan-Maxwell		
T	Gas Channel Energy Balance	Equation A.4	
p	$p = p_a$	Darcy's Law	
N_{HF}	$\nabla \cdot N_{HF} = 0$		
c_{HF}	Fick's Law		

Anode Catalyst Layer

Variable	Anode Diffusion Media / Anode Catalyst Layer Boundary	Anode Catalyst Layer	Anode Catalyst Layer / Membrane Boundary
i_1	$\nabla \cdot i_1 + \nabla \cdot i_2 = 0$		
Φ_1	$i_1 = -\sigma \nabla \Phi_1$		
i_2	$i_2 = 0$	Equation A.7	
Φ_2	$i_2 = -\kappa \nabla \Phi_2 - \frac{\kappa \xi}{F} \nabla \mu_w$		
μ_w	$N_{w,M} = \frac{\kappa \xi}{F} \nabla \Phi_2 + \left(\alpha + \frac{\kappa \xi^2}{F^2} \right) \nabla \mu_w$		
$N_{w,M}$	$N_{w,M} = 0$	Equation A.8	
N_{O_2}	Equation A.9		
N_{N_2}	$\nabla \cdot N_{N_2} = 0$		
N_w	Equation A.3		
N_{H_2}	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$		
y_{O_2}	Stefan-Maxwell		
y_{N_2}	Stefan-Maxwell		
y_w	Stefan-Maxwell		
y_{H_2}	Stefan-Maxwell		
T	Equation A.4		
p	Darcy’s Law		
N_{HF}	$\nabla \cdot N_{HF} = 0$		
c_{HF}	Fick’s Law		

Membrane

Variable	**Anode Catalyst Layer / Membrane Boundary**	**Membrane**	**Membrane / Cathode Catalyst Layer Boundary**
i_1	$\nabla \cdot i_1 + \nabla \cdot i_2 = 0$	$i_1 = 0$	$\nabla \cdot i_1 + \nabla \cdot i_2 = 0$
Φ_1	$i_1 = -\sigma \nabla \Phi_1$	$\Phi_1 = 0$	$i_1 = -\sigma \nabla \Phi_1$
i_2	Equation A.7	$i_2 = 0$	Equation A.7
Φ_2	$i_2 = -\kappa \nabla \Phi_2 - \frac{\kappa \xi}{F} \nabla \mu_w$		
μ_w	$N_{w,M} = \frac{\kappa \xi}{F} \nabla \Phi_2 + \left(\alpha + \frac{\kappa \xi^2}{F^2} \right) \nabla \mu_w$		
$N_{w,M}$	Equation A.8		
N_{O_2}	Equation A.9	$\nabla \cdot N_{O_2} = 0$	Equation A.9
N_{N_2}	$\nabla \cdot N_{N_2} = 0$		
N_w	Equation A.3		
N_{H_2}	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$	$\nabla \cdot N_{H_2} = 0$	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$
y_{O_2}	Stefan-Maxwell		
y_{N_2}	Stefan-Maxwell		
y_w	Stefan-Maxwell		
y_{H_2}	Stefan-Maxwell		
T	Equation A.4		
p	Darcy's Law		
N_{HF}	$\nabla \cdot N_{HF} = 0$		
c_{HF}	Fick's Law		

Cathode Catalyst Layer

	Membrane / Cathode Catalyst Layer Boundary	Cathode Catalyst Layer	Cathode Catalyst Layer / Cathode Diffusion Media Boundary
i_1	$\nabla \cdot i_1 + \nabla \cdot i_2 = 0$		
Φ_1	$i_1 = -\sigma \nabla \Phi_1$		
i_2	Equation A.7		
Φ_2	$i_2 = -\kappa \nabla \Phi_2 - \frac{\kappa \xi}{F} \nabla \mu_w$		$\Phi_2 = 0$
μ_w	$N_{w,M} = \frac{\kappa \xi}{F} \nabla \Phi_2 + \left(\alpha + \frac{\kappa \xi^2}{F^2} \right) \nabla \mu_w$		$N_{w,mem} = 0$
$N_{w,M}$	Equation A.8		
N_{O_2}	Equation A.9		
N_{N_2}	$\nabla \cdot N_{N_2} = 0$		
N_w	Equation A.3		
N_{H_2}	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$		$\nabla \cdot N_{H_2} = 0$
y_{O_2}	Stefan-Maxwell		
y_{N_2}	Stefan-Maxwell		
y_w	Stefan-Maxwell		
y_{H_2}	Stefan-Maxwell		
T	Equation A.4		
p	Darcy's Law		
N_{HF}	$\nabla \cdot N_{HF} = 0$	At midpoint of CL: $N_{HF} = 0$	$\nabla \cdot N_{HF} = 0$
c_{HF}	Fick's Law	At midpoint of CL: $\frac{dc_{HF}}{dx} = 0$	Fick's Law

Cathode Diffusion Media

	Cathode Catalyst Layer / Cathode Diffusion Media Boundary	**Cathode Diffusion Media**	**Cathode Diffusion Media / Cathode Gas Channel Boundary**
i_1	$\nabla \cdot i_1 + \nabla \cdot i_2 = 0$		
Φ_1	$i_1 = -\sigma \nabla \Phi_1$		$\Phi_1 = \Phi_{cell}$
i_2	Equation A.7	$i_2 = 0$	
Φ_2	$\Phi_2 = 0$		
μ_w	$N_{w,M} = 0$	$\mu_w = 0$	
$N_{w,M}$	Equation A.8	$N_{w,mem} = 0$	
N_{O_2}	Equation A.9	$\nabla \cdot N_{O_2} = 0$	
N_{N_2}	$\nabla \cdot N_{N_2} = 0$		
N_w	Equation A.3	$\nabla \cdot N_w = 0$	Gas Channel Mass Balance
N_{H_2}	$\nabla \cdot N_{H_2} = 0$		
y_{O_2}	Stefan-Maxwell		$\sum_i y_i = 1$
y_{N_2}	Stefan-Maxwell		Gas Channel Mass Balance
y_w	Stefan-Maxwell		Gas Channel Mass Balance
y_{H_2}	Stefan-Maxwell		$N_{H_2} = 0$
T	Equation A.4		Gas Channel Energy Balance
p	Darcy’s Law		
N_{HF}	$\nabla \cdot N_{HF} = 0$		
c_{HF}	Fick’s Law		$c_{HF} = 0$

B.2 Multiphase Performance Model with Empirical Chemical Degradation

Equations are the same as B.1 except for those shown.

Anode Diffusion Media

Variable	**Anode Gas Channel / Anode Diffusion Media Boundary**	**Anode Diffusion Media**	**Anode Diffusion Media / Anode Catalyst Layer Boundary**
Φ_2	$\Phi_2 = 0$		Equation A.10
μ_w	$\mu_w = 0$		$\mu_w = \bar{V}_w p_{L,M}$
$N_{w,M}$	$N_{w,M} = 0$		
N_w	Gas Channel Mass Balance	$\nabla \cdot N_w = 0$	Equation A.13
y_{O_2}	$y_{O_2} = 0$	Stefan-Maxwell	
y_{N_2}	$y_{N_2} = 0$	Stefan-Maxwell	
y_{H_2}	$\sum_i y_i = 1$		
T	Gas Channel Energy Balance	Equation A.4	
p_g	$p = p_a$	Darcy's Law	
N_{HF}	$\nabla \cdot N_{HF} = 0$		
c_{HF}	Fick's Law		
p_L	$p_l = N_{w,L} + (p_L - p_{thru})(\tanh(p_L - p_{thru}) + 1)$	Darcy's Law	
$p_{L,M}$	$p_{L,M} = 0$		Equation A.11
$N_{w,L}$	$\nabla \cdot N_L = k_{M,L}\left(p_{L,M} - p_L\right) - \frac{k_{evap}\left(p_g y_w - p_w^{sat}\right)}{RT}$		

Anode Catalyst Layer

Variable	Anode Diffusion Media / Anode Catalyst Layer Boundary	Anode Catalyst Layer	Anode Catalyst Layer / Membrane Boundary
Φ_2	Equation A.10		
μ_w	$\mu_w = \bar{V}_w p_{L,M}$		
$N_{w,M}$	$N_{w,M} = 0$	Equation A.12	
N_w	Equation A.13		
y_{O_2}	Stefan-Maxwell		$N_{O_2} = -\psi_{O_2} \nabla y_{O_2}$
y_{N_2}	Stefan-Maxwell		$N_{N_2} = -\psi_{O_2} \nabla y_{N_2}$
y_{H_2}	$\sum_i y_i = 1$		
T	Equation A.4		
p_g	Darcy's Law		
N_{HF}	$\nabla \cdot N_{HF} = 0$		
c_{HF}	Fick's Law		
p_L	Darcy's Law		
$p_{L,M}$	Equation A.11		
$N_{w,L}$	$\nabla \cdot N_L = k_{M,L}\left(p_{L,M} - p_L\right) - \frac{k_{evap}\left(p_g y_w - p_w^{sat}\right)}{RT}$		

Membrane

Variable	Anode Catalyst Layer / Membrane Boundary	Membrane	Membrane / Cathode Catalyst Layer Boundary
Φ_2	Equation A.10		
μ_w	$\mu_w = \bar{V}_w p_{L,M}$		
$N_{w,M}$	Equation A.12		
N_w	Equation A.13		
y_{O_2}	$N_{O_2} = -\psi_{O_2} \nabla y_{O_2}$		$\sum_i y_i = 1$
y_{N_2}	$N_{N_2} = -\psi_{O_2} \nabla y_{N_2}$		Stefan-Maxwell
y_{H_2}	$\sum_i y_i = 1$	$N_{H_2} = -\psi_{H_2} \nabla y_{H_2}$	
T	Equation A.4		
p_g	Darcy's Law	$p_g = 0$	Darcy's Law
N_{HF}	$\nabla \cdot N_{HF} = 0$		
c_{HF}	Fick's Law		
p_L	Darcy's Law	$p_L = 0$	Darcy's Law
$p_{L,M}$	Equation A.11		$p_{L,M} = 0$
$N_{w,L}$	$\nabla \cdot N_L = k_{M,L}\left(p_{L,M} - p_L\right) - \frac{k_{evap}\left(p_g y_w - p_w^{sat}\right)}{RT}$	$N_{w,L} = 0$	$\nabla \cdot N_L = k_{M,L}\left(p_{L,M} - p_L\right) - \frac{k_{evap}\left(p_g y_w - p_w^{sat}\right)}{RT}$

Cathode Catalyst Layer

	Membrane / Cathode Catalyst Layer Boundary	**Cathode Catalyst Layer**	**Cathode Catalyst Layer / Cathode Diffusion Media Boundary**
Φ_2	Equation A.10	Equation A.10	$\Phi_2 = 0$
μ_w	$\mu_w = \bar{V}_w p_{L,M}$	$\mu_w = \bar{V}_w p_{L,M}$	$\mu_w = \bar{V}_w p_{L,M}$
$N_{w,M}$	Equation A.12	Equation A.12	Equation A.12
N_w	Equation A.13	Equation A.13	Equation A.13
y_{O_2}	$\sum_i y_i = 1$	$\sum_i y_i = 1$	$\sum_i y_i = 1$
y_{N_2}	Stefan-Maxwell	Stefan-Maxwell	Stefan-Maxwell
y_{H_2}	$N_{H_2} = -\psi_{H_2} \nabla y_{H_2}$	Stefan-Maxwell	Stefan-Maxwell
T	Equation A.4	Equation A.4	Equation A.4
p_g	Darcy's Law	Darcy's Law	Darcy's Law
N_{HF}	$\nabla \cdot N_{HF} = 0$	At midpoint of CL: $N_{HF} = 0$	$\nabla \cdot N_{HF} = 0$
c_{HF}	Fick's Law	At midpoint of CL: $\frac{dc_{HF}}{dx} = 0$	Fick's Law
p_L	Darcy's Law	Darcy's Law	Darcy's Law
$p_{L,M}$	$p_{L,M} = 0$	$p_{L,M} = 0$	$p_{L,M} = 0$
$N_{w,L}$	$\nabla \cdot N_L = k_{M,L}\left(p_{L,M} - p_L\right) - \frac{k_{evap}\left(p_g y_w - p_w^{sat}\right)}{RT}$	$\nabla \cdot N_L = k_{M,L}\left(p_{L,M} - p_L\right) - \frac{k_{evap}\left(p_g y_w - p_w^{sat}\right)}{RT}$	$\nabla \cdot N_L = k_{M,L}\left(p_{L,M} - p_L\right) - \frac{k_{evap}\left(p_g y_w - p_w^{sat}\right)}{RT}$

Cathode Diffusion Media

	Cathode Catalyst Layer / Cathode Diffusion Media Boundary	Cathode Diffusion Media	Cathode Diffusion Media / Cathode Gas Channel Boundary
Φ_2	$\Phi_2 = 0$		
μ_w	$\mu_w = \bar{V}_w p_{L,M}$	$\mu_w = 0$	
$N_{w,M}$	Equation A.12	$N_{w,mem} = 0$	
N_{N_2}	$\nabla \cdot N_{N_2} = 0$		
N_w	Equation A.13	$\nabla \cdot N_w = 0$	Gas Channel Mass Balance
y_{O_2}	$\sum_i y_i = 1$		
y_{N_2}	Stefan-Maxwell		Gas Channel Mass Balance
y_{H_2}	Stefan-Maxwell		$N_{H_2} = 0$
T	Equation A.4		Gas Channel Energy Balance
p_g	Darcy's Law		
N_{HF}	$\nabla \cdot N_{HF} = 0$		
c_{HF}	Fick's Law		$c_{HF} = 0$
p_L	Darcy's Law		$p_l = N_{w,L} + (p_L - p_{thru})(\tanh(p_L - p_{thru}) + 1)$
$p_{L,M}$	$p_{L,M} = 0$		
$N_{w,L}$	$\nabla \cdot N_L = k_{M,L}\left(p_{L,M} - p_L\right) - \frac{k_{evap}\left(p_g y_w - p_w^{sat}\right)}{RT}$		

B.3 Microkinetic Chemical Degradation Model and Cerium Effects

Anode Diffusion Media

Variable	Anode Gas Channel / Anode Diffusion Media Boundary	Anode Diffusion Media	Anode Diffusion Media / Anode Catalyst Layer Boundary
i_1	$\nabla \cdot i_1 + \nabla \cdot i_2 = 0$		
Φ_1	$\Phi_1 = 0$	$i_1 = -\sigma\nabla\Phi_1$	$\Phi_1 = 0$
i_2	$i_2 = 0$		
Φ_2	$\Phi_2 = 0$		Equation A.1
μ_w	$\mu_w = 0$		Equation A.2
$N_{w,M}$	$N_{w,M} = 0$		
N_{O_2}	$N_{O_2} = 0$		
N_{N_2}	$N_{N_2} = 0$		
N_w	Gas Channel Mass Balance	$\nabla \cdot N_w = 0$	Equation A.3
N_{H_2}	$\nabla \cdot N_{H_2} = 0$		
y_{O_2}	Stefan-Maxwell		
y_{N_2}	$y_{N_2} = 0$		
y_w	$\nabla \cdot N_w = 0$	Stefan-Maxwell	
y_{H_2}	$\sum_i y_i = 1$		
T	Gas Channel Energy Balance	Equation A.4	
p	$p = p_a$	Darcy's Law	
ℓ	$\ell = 0$		
τ	$\tau = 0$		
f_{Ce}	$f_{Ce} = 0$		$N_{Ce} = 0$
μ_{Ce}	$\mu_{Ce} = 0$		Equation A.5
N_{Ce}	$N_{Ce} = 0$		Equation A.6
n_{Ce}	$n_{Ce} = 0$		
$N_{H_2O_2}$	$c_{H_2O_2} = 0$	$\nabla \cdot N_{H_2O_2} = 0$	
$c_{H_2O_2}$	Fick's Law		

$c_{R_f SO_3}$	$c_{R_f SO_3} = 0$	
N_{HF}	$c_{HF} = 0$	$\nabla \cdot N_{HF} = 0$
c_{HF}	Fick's Law	
$c_{OH^\bullet}$	$c_{OH^\bullet} = 0$	
$c_{R_f \alpha O^\bullet}$	$c_{R_f \alpha O^\bullet} = 0$	
$c_{R_f \beta O^\bullet}$	$c_{R_f \beta O^\bullet} = 0$	
$c_{R_f COOH}$	$c_{R_f COOH} = 0$	

Anode Catalyst Layer

Variable	**Anode Diffusion Media / Anode Catalyst Layer Boundary**	**Anode Catalyst Layer**	**Anode Catalyst Layer / Membrane Boundary**
i_1	$\nabla \cdot i_1 + \nabla \cdot i_2 = 0$		$i_1 = 0$
Φ_1	$i_1 = -\sigma \nabla \Phi_1$		
i_2	$i_2 = 0$	Equation A.7	
Φ_2	Equation A.1		
μ_w	Equation A.2		
$N_{w,M}$	Equation A.8		
N_{O_2}	Equation A.9		
N_{N_2}	$N_{N_2} = 0$		
N_w	Equation A.3		
N_{H_2}	$\nabla \cdot N_{H_2} = 0$	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$	
y_{O_2}	Stefan-Maxwell		$N_{O_2} = -\psi_{O_2} \nabla y_{O_2}$
y_{N_2}	$y_{N_2} = 0$		
y_w	Stefan-Maxwell		
y_{H_2}	$\sum_i y_i = 1$		
T	Equation A.4		
p	Darcy's Law		
ℓ	$\ell = 0$		$\ell = \tau$
τ	$\tau = 0$	$\frac{d\tau}{dx} = 1 + 0.36\lambda \frac{\bar{V}_0}{\bar{V}_{mem}}$	
f_{Ce}	$N_{Ce} = 0$	$\frac{dc_{Ce}}{dt} = \nabla \cdot N_{Ce}$	
μ_{Ce}	Equation A.5		
N_{Ce}	Equation A.6		
n_{Ce}	$n_{Ce} = 0$	$\frac{dn_{Ce}}{dx} = c_{Ce}$	
$N_{H_2O_2}$	$\nabla \cdot N_{H_2O_2} = 0$		
$c_{H_2O_2}$	Fick's Law		

$c_{R_fSO_3}$	$c_{R_fSO_3} = 0$	$N_{R_fSO_3} = 0$
N_{HF}	$\nabla \cdot N_{HF} = 0$	
c_{HF}	Fick's Law	
$c_{OH^{\bullet}}$	$N_{OH^{\bullet}} = 0$	
$c_{R_f\alpha O^{\bullet}}$	$N_{R_f\alpha O^{\bullet}} = 0$	
$c_{R_f\beta O^{\bullet}}$	$N_{R_f\beta O^{\bullet}} = 0$	
c_{R_fCOOH}	$N_{R_fCOOH} = 0$	

Membrane

Variable	Anode Catalyst Layer / Membrane Boundary	Membrane	Membrane / Cathode Catalyst Layer Boundary
i_1	$i_1 = 0$		
Φ_1	$i_1 = -\sigma\nabla\Phi_1$	$\Phi_1 = 0$	$i_1 = -\sigma\nabla\Phi_1$
i_2	Equation A.7	$i_2 = 0$	Equation A.7
Φ_2	Equation A.1		
μ_w	Equation A.2		
$N_{w,M}$	Equation A.8	$\nabla \cdot N_{w,M} = 0$	Equation A.8
N_{O_2}	Equation A.9	$\nabla \cdot N_{O_2} = 0$	Equation A.9
N_{N_2}	$N_{N_2} = 0$		
N_w	Equation A.3	$N_w = 0$	Equation A.3
N_{H_2}	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$	$\nabla \cdot N_{H_2} = 0$	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$
y_{O_2}	$N_{O_2} = -\psi_{O_2}\nabla y_{O_2}$		$\sum_i y_i = 1$
y_{N_2}	$y_{N_2} = 0$		Stefan-Maxwell
y_w	Stefan-Maxwell	$y_w = 0$	Stefan-Maxwell
y_{H_2}	$\sum_i y_i = 1$	$N_{H_2} = -\psi_{H_2}\nabla y_{H_2}$	
T	Equation A.4		
p	Darcy's Law	$p = 0$	Darcy's Law
ℓ	$\ell = \tau$	$\frac{d\ell}{dx} = 0$	
τ	$\frac{d\tau}{dx} = 1 + 0.36\lambda \frac{\bar{V}_0}{\bar{V}_{mem}}$		$\tau = 0$
f_{Ce}	$\frac{dc_{Ce}}{dt} = \nabla \cdot N_{Ce}$		
μ_{Ce}	Equation A.5		
N_{Ce}	Equation A.6		
n_{Ce}	$\frac{dn_{Ce}}{dx} = c_{Ce}$		
$N_{H_2O_2}$	$\nabla \cdot N_{H_2O_2} = 0$		
$c_{H_2O_2}$	Fick's Law		

$c_{R_fSO_3}$	$N_{R_fSO_3} = 0$
N_{HF}	$\nabla \cdot N_{HF} = 0$
c_{HF}	Fick’s Law
$c_{OH^\bullet}$	$N_{OH^\bullet} = 0$
$c_{R_f\alpha O^\bullet}$	$N_{R_f\alpha O^\bullet} = 0$
$c_{R_f\beta O^\bullet}$	$N_{R_f\beta O^\bullet} = 0$
c_{R_fCOOH}	$N_{R_fCOOH} = 0$

Cathode Catalyst Layer

	Membrane / Cathode Catalyst Layer Boundary	**Cathode Catalyst Layer**	**Cathode Catalyst Layer / Cathode Diffusion Media Boundary**
i_1	$i_1 = 0$	$\nabla \cdot i_1 + \nabla \cdot i_2 = 0$	
Φ_1	$i_1 = -\sigma \nabla \Phi_1$		
i_2	Equation A.7		
Φ_2	Equation A.1		$\Phi_2 = 0$
μ_w	Equation A.2		$N_{w,mem} = 0$
$N_{w,M}$	Equation A.8		
N_{O_2}	Equation A.9		
N_{N_2}	$N_{N_2} = 0$	$\nabla \cdot N_{N_2} = 0$	
N_w	Equation A.3		
N_{H_2}	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$		$N_{H_2} = 0$
y_{O_2}	$N_{O_2} = -\psi_{O_2} \nabla y_{O_2}$	$\sum_i y_i = 1$	
y_{N_2}	Stefan-Maxwell		
y_w	Stefan-Maxwell		
y_{H_2}	$N_{H_2} = -\psi_{H_2} \nabla y_{H_2}$	Stefan-Maxwell	
T	Equation A.4		
p	Darcy's Law		
ℓ	$\frac{d\ell}{dx} = 0$	$\ell = 0$	
τ	$\tau = 0$		
f_{Ce}	$\frac{dc_{Ce}}{dt} = \nabla \cdot N_{Ce}$		
μ_{Ce}	Equation A.5		$n_{Ce} = f_{Ce,0} \left(\frac{\rho_M}{EW}\right) \ell_{M,0}$
N_{Ce}	Equation A.6		$N_{Ce} = 0$
n_{Ce}	$\frac{dn_{Ce}}{dx} = c_{Ce}$		
$N_{H_2O_2}$	$\nabla \cdot N_{H_2O_2} = 0$		

$c_{H_2O_2}$	Fick's Law
$c_{R_fSO_3}$	$N_{R_fSO_3} = 0$
N_{HF}	$\nabla \cdot N_{HF} = 0$
c_{HF}	Fick's Law
$c_{OH^\bullet}$	$N_{OH^\bullet} = 0$
$c_{R_f\alpha O^\bullet}$	$N_{R_f\alpha O^\bullet} = 0$
$c_{R_f\beta O^\bullet}$	$N_{R_f\beta O^\bullet} = 0$
c_{R_fCOOH}	$N_{R_fCOOH} = 0$

Cathode Diffusion Media

	Cathode Catalyst Layer / Cathode Diffusion Media Boundary	**Cathode Diffusion Media**	**Cathode Diffusion Media / Cathode Gas Channel Boundary**
i_1	$\nabla \cdot i_1 + \nabla \cdot i_2 = 0$		
Φ_1	$i_1 = -\sigma \nabla \Phi_1$		$\Phi_1 = \Phi_{cell}$
i_2	Equation A.7	$i_2 = 0$	
Φ_2	$\Phi_2 = 0$		
μ_w	$N_{w,M} = 0$	$\mu_w = 0$	
$N_{w,M}$	Equation A.8	$N_{w,mem} = 0$	
N_{O_2}	Equation A.9	$\nabla \cdot N_{O_2} = 0$	
N_{N_2}	$\nabla \cdot N_{N_2} = 0$		
N_w	Equation A.3	$\nabla \cdot N_w = 0$	Gas Channel Mass Balance
N_{H_2}	$N_{H_2} = 0$		
y_{O_2}	$\sum_i y_i = 1$		
y_{N_2}	Stefan-Maxwell		Gas Channel Mass Balance
y_w	Stefan-Maxwell		$\nabla \cdot N_w = 0$
y_{H_2}	Stefan-Maxwell		
T	Equation A.4		Gas Channel Energy Balance
p	Darcy's Law		$P = P_c$
ℓ	$\ell = 0$		
τ	$\tau = 0$		
f_{Ce}	$\frac{dc_{Ce}}{dt} = \nabla \cdot N_{Ce}$	$f_{Ce} = 0$	
μ_{Ce}	$n_{Ce} = f_{Ce,0}\left(\frac{\rho_M}{EW}\right)\ell_{M,0}$	$\mu_{Ce} = 0$	
N_{Ce}	$N_{Ce} = 0$		
n_{Ce}	$\frac{dn_{Ce}}{dx} = c_{Ce}$	$n_{Ce} = 0$	
$N_{H_2O_2}$	$\nabla \cdot N_{H_2O_2} = 0$		
$c_{H_2O_2}$	Fick's Law		$c_{H_2O_2} = 0$

$c_{R_fSO_3}$	$c_M = \dfrac{(1-f_w)\rho_M}{EW}$	$c_{R_fSO_3} = 0$
N_{HF}	$\nabla \cdot N_{HF} = 0$	
c_{HF}	Fick's Law	$c_{HF} = 0$
$c_{OH^\bullet}$	$c_{OH^\bullet} = 0$	
$c_{R_f\alpha O^\bullet}$	$c_{R_f\alpha O^\bullet} = 0$	
$c_{R_f\beta O^\bullet}$	$c_{R_f\beta O^\bullet} = 0$	
c_{R_fCOOH}	$c_{R_fCOOH} = 0$	

B.4 Mechanical Degradation Model with Microkinetic Chemical Degradation Model and Cerium Effects

Equations are the same as B.3 except for those shown.

Anode Diffusion Media

Variable	Anode Gas Channel / Anode Diffusion Media Boundary	Anode Diffusion Media	Anode Diffusion Media / Anode Catalyst Layer Boundary
N_{O_2}	$N_{O_2} = 0$	$\nabla \cdot N_{O_2} = 0$	
N_{N_2}	Gas Channel Mass Balance	$\nabla \cdot N_{N_2} = 0$	
N_w	Gas Channel Mass Balance	Equation A.3	
N_{H_2}	$\sum_i y_i = 1$	$\nabla \cdot N_{H_2} = 0$	
y_{O_2}	Stefan-Maxwell		
y_{N_2}	Stefan-Maxwell	$\sum_i y_i = 1$	
y_w	Stefan-Maxwell		
y_{H_2}	Stefan-Maxwell		
p	$p = p_a$	Darcy's Law	

Anode Catalyst Layer

Variable	Anode Diffusion Media / Anode Catalyst Layer Boundary	Anode Catalyst Layer	Anode Catalyst Layer / Membrane Boundary
N_{O_2}	Equation A.9		
N_{N_2}	$\nabla \cdot N_{N_2} = 0$		
N_w	Equation A.2		
N_{H_2}	$\nabla \cdot N_{H_2} = 0$	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$	
y_{O_2}	Stefan-Maxwell		
y_{N_2}	$\sum_i y_i = 1$		
y_w	Stefan-Maxwell		
y_{H_2}	Stefan-Maxwell		
p	Darcy's Law		

Membrane

Variable	Anode Catalyst Layer / Membrane Boundary	Membrane	Membrane / Cathode Catalyst Layer Boundary
N_{O_2}	Equation A.9		
N_{N_2}	$\nabla \cdot N_{N_2} = 0$		
N_w	Equation A.3		
N_{H_2}	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$		
y_{O_2}	Stefan-Maxwell		
y_{N_2}	$\sum_i y_i = 1$		
y_w	Stefan-Maxwell		
y_{H_2}	Stefan-Maxwell		
p	Darcy's Law		

Cathode Catalyst Layer

	Membrane / Cathode Catalyst Layer Boundary	Cathode Catalyst Layer	Cathode Catalyst Layer / Cathode Diffusion Media Boundary
$N_{w,M}$	Equation A.8		
N_{O_2}	Equation A.9		
N_{N_2}	$\nabla \cdot N_{N_2} = 0$		
N_w	Equation A.3		
N_{H_2}	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$		
y_{O_2}	Stefan-Maxwell		$\sum_i y_i = 1$
y_{N_2}	$\sum_i y_i = 1$		
y_w	Stefan-Maxwell		
y_{H_2}	Stefan-Maxwell		
p	Darcy's Law		

Cathode Diffusion Media

	Cathode Catalyst Layer / Cathode Diffusion Media Boundary	Cathode Diffusion Media	Cathode Diffusion Media / Cathode Gas Channel Boundary
N_{O_2}	Equation A.9	$\nabla \cdot N_{O_2} = 0$	
N_{N_2}	$\nabla \cdot N_{N_2} = 0$		
N_w	Equation A.3		
N_{H_2}	$\nabla \cdot N_{H_2} = -\frac{1}{2F} i_{HOR}$	$\nabla \cdot N_{H_2} = 0$	
y_{O_2}	$\sum_i y_i = 1$		
y_{N_2}	$\sum_i y_i = 1$		Gas Channel Mass Balance
y_w	Stefan-Maxwell		Gas Channel Mass Balance
y_{H_2}	Stefan-Maxwell		$N_{H_2} = 0$
p	Darcy's Law		

Appendix B – MATLAB Code

All codes are written and complied in MATLAB R2020a.

B.1 BAND(J) and Finite Volume Method Codes

```
function C = autoband(n,nj,C,Cp,dC,params)

J = zeros(n*nj,n*nj); % block tridiagonal matrix
b = zeros(n*nj,1);

for j = 1:nj
    A = zeros(n,n);      % matrix of dG/dC at j-1
    B = zeros(n,n);      % matrix of dG/dC at j
    D = zeros(n,n);      % matrix of dG/dC at j+1

    % initialize G (k = 1, dC = 0)
    % matrix of governing equations
    G = eqn(j,j,1,0,C,Cp,params);

    % generate A,B,D matrices
    for k = 1:n
        eq = eqn(j,j,k,dC(k),C,Cp,params);
        B(:,k) = -(eq-G)./dC(k);
        if j > 1
            eq = eqn(j,j-1,k,dC(k),C,Cp,params);
            A(:,k) = -(eq-G)./dC(k);
        end
        if j < nj
            eq = eqn(j,j+1,k,dC(k),C,Cp,params);
            D(:,k) = -(eq-G)./dC(k);
        end
        % construct tridiagonal matrix
        for m = 1:n
            J((m-1)*nj+j,(k-1)*nj+j) = B(m,k);
            if j > 1
                J((m-1)*nj+j,(k-1)*nj+j-1) = A(m,k);
            end
            if j < nj
                J((m-1)*nj+j,(k-1)*nj+j+1) = D(m,k);
            end
        end
        % construct solution vector
        b((k-1)*nj+j) = G(k);
    end
end

Js = sparse(J);
U = Js\b;
C = reshape(U,nj,n);
end
```

```
function [C,iflag] = steady_state(params,itmax,C_ss0)
C = C_ss0; % intialize
jcount = 0; % current iteration

% Delta C = small variation in value of C
dC = 1e-8*[ones(1,5) 1e-4*ones(1,5) ones(1,9)...
    1 1e-4*ones(1,4) 1 1e-4*ones(1,6)];
rtol = 1e-6; atol = 1e-9; kerr = 1; kerrg = 1;
iflag = 0; % return flag if simulation does not converge
n = params(1); nj = params(14);
while (kerr == 1 || kerrg == 1) && jcount < itmax
    jcount = jcount+1;    % update iteration
    CC = C;               % initialize CC
    C = autoband(n,nj,C,[],dC,params);
    C = shoehorn(n,C,CC,params);
    kerr = 0; kerrg = 0;
    for j = 1:nj
        for i = 1:n
            if kerr == 0 && kerrg == 0
                if abs(C(j,i)) > rtol*abs(CC(j,i))
                    kerr = 1;
                end
                if kerr == 1 && abs(abs(C(j,i))<atol)
                    kerr = 0;
                end
            end
        end
        for i = 1:n
            C(j,i) = CC(j,i)+C(j,i);
        end
    end
    if kerr == 0 && kerrg == 0
        fprintf("\n Simulation has converged\n")
    end
    if jcount >= itmax
        iflag = 1;
    end
end
end
```

```
function [C,iflag] = transient(params,itmax,Cp)
C = Cp; % intial condition
jcount = 0;          % current iteration
% Delta C = small variation in value of C
dC = 1e-8*[ones(1,5) 1e-4*ones(1,5) ones(1,9)...
    1 1e-4*ones(1,4) 1 1e-4*ones(1,6)];
rtol = 1e-6; atol = 1e-9; kerr = 1; kerrg = 1;
iflag = 0; % return flag if simulation does not converge
n = params(1); nj = params(14);
while (kerr == 1 || kerrg == 1) && jcount < itmax
    jcount = jcount+1;   % update iteration
    CC = C;              % initialize CC
    C = autoband(n,nj,C,Cp,dC,params);
    C = shoehorn(n,C,CC,params);
    kerr = 0;  kerrg = 0;
    for j = 1:nj
        for i = 1:n
            if kerr == 0 && kerrg == 0
                if abs(C(j,i)) > rtol*abs(CC(j,i))
                    kerr = 1;
                end
                if kerr == 1 && abs(abs(C(j,i))<atol)
                    kerr = 0;
                end
            end
        end
        for i = 1:n
            C(j,i) = CC(j,i)+C(j,i);
        end
    end
    if jcount >= itmax
        iflag = 1;
    end
end
end
```

```
function [RH1,RH2] = RH_cycle(t,dt,rh1,rh2,rh0,t_ramp,...
    t_hold,t_i,h_cycle)

t_cycle = 2*t_ramp + 2*t_hold; % total time for cycle (s)
cycle = ceil((t-t_i)/t_cycle);

if t <= t_i
    if t < t_i
        RH1 = rh1;
        RH2 = rh2;
    else
        RH1 = rh1+dt*(h_cycle/t_ramp);
        RH2 = rh2+dt*(h_cycle/t_ramp);
    end
elseif t > (cycle-1)*t_cycle+t_i &&...
        t < (cycle-1)*t_cycle+t_ramp+t_i
        RH1 = rh1+dt*(h_cycle/t_ramp);
        RH2 = rh2+dt*(h_cycle/t_ramp);

elseif t >= (cycle-1)*t_cycle+t_ramp+t_i &&...
        t < (cycle-1)*t_cycle+t_ramp+t_hold+t_i
        RH1 = rh0+h_cycle;
        RH2 = rh0+h_cycle;

elseif t >= (cycle-1)*t_cycle+t_ramp+t_hold+t_i &&...
        t < (cycle-1)*t_cycle+2*t_ramp+t_hold+t_i
        RH1 = rh1-dt*(h_cycle/t_ramp);
        RH2 = rh2-dt*(h_cycle/t_ramp);

elseif t >= (cycle-1)*t_cycle+2*t_ramp+t_hold+t_i &&...
        t <= cycle*t_cycle+t_i
    if t < cycle*t_cycle+t_i
        RH1 = rh0;
        RH2 = rh0;
    else
        RH1 = rh1+dt*(h_cycle/t_ramp);
        RH2 = rh2+dt*(h_cycle/t_ramp);
    end
end

if RH1 < rh0
    RH1 = rh0;
end
if RH2 < rh0
    RH2 = rh0;
end
if RH1 > rh0+h_cycle
    RH1 = rh0+h_cycle;
end
if RH2 > rh0+h_cycle
    RH2 = rh0+h_cycle;
end
end
```

```
function V = V_cycle(t,dt,v,v0,t_ramp,t_hold,t_i,h_cycle)
t_cycle = 2*t_ramp + 2*t_hold; % total time for cycle (s)
cycle = ceil((t-t_i)/t_cycle);
if t <= t_i
    if t < t_i
        V = v;
    else
        V = v+dt*(h_cycle/t_ramp);
    end
elseif t > (cycle-1)*t_cycle+t_i &&...
        t < (cycle-1)*t_cycle+t_ramp+t_i
        V = v+dt*(h_cycle/t_ramp);

elseif t >= (cycle-1)*t_cycle+t_ramp+t_i &&...
        t < (cycle-1)*t_cycle+t_ramp+t_hold+t_i
        V = v+h_cycle;

elseif t >= (cycle-1)*t_cycle+t_ramp+t_hold+t_i &&...
        t < (cycle-1)*t_cycle+2*t_ramp+t_hold+t_i
        V = v-dt*(h_cycle/t_ramp);

elseif t >= (cycle-1)*t_cycle+2*t_ramp+t_hold+t_i &&...
        t <= cycle*t_cycle+t_i
    if t < cycle*t_cycle+t_i
        V = v0;
    else
        V = v+dt*(h_cycle/t_ramp);
    end
end

if V < v0 && h_cycle > 0
    V = v0;
elseif V > v0 && h_cycle < 0
    V = v0;
end
if V > v0+h_cycle && h_cycle > 0
    V = v0+h_cycle;
elseif V < v0+h_cycle && h_cycle < 0
    V = v0+h_cycle;
end
end
```

```
function dx = mesh(j,C,iregion,params)
% This function determines the mesh spacing
iL = 17;
L = params(15:19); bound = params(9:14);
if iregion == 3
    dx = C(j,iL)/(bound(iregion+1)-bound(iregion));
else
    dx = L(iregion)/(bound(iregion+1)-bound(iregion));
end
end
```

```
function iregion = region(mode,j,bound)
if mode == 1 % box to the left of j
    if j <= bound(2)
        iregion = 1;
    elseif j > bound(2) && j <= bound(3)
        iregion = 2;
    elseif j > bound(3) && j <= bound(4)
        iregion = 3;
    elseif j > bound(4) && j <= bound(5)
        iregion = 4;
    else
        iregion = 5;
    end
else  % box to the right of j
    if j < bound(2)
        iregion = 1;
    elseif j >= bound(2) && j < bound(3)
        iregion = 2;
    elseif j >= bound(3) && j < bound(4)
        iregion = 3;
    elseif j >= bound(4) && j < bound(5)
        iregion = 4;
    else
        iregion = 5;
    end
end
end
```

```
function C = shoehorn(n,C,CC,params)
nspecies = params(3); bound = params(9:14); nj = bound(6);
iv1 = 2; iv2 = 4; imuw = 5; iyO2 = 11; ipg = 16; ifCe = 19; inCe = 22;
imuCe = 20; icRfSO3 = 25; icH2O2 = 24; icHF = 27; icRfalphaO = 29;
icRfbetaO = 30;
fCe0 = params(41); vmax = 0.05;
for j = 1:nj
    for i = 1:n
        CN(i) = CC(j,i)+C(j,i);
    end
    if abs(C(j,iv1)) > vmax
        C(j,iv1) = vmax*sign(C(j,iv1));
    end
    if abs(C(j,iv2)) > vmax
        C(j,iv2) = vmax*sign(C(j,iv2));
    end
    for ii = [iyO2:iyO2+nspecies-1,ifCe]
        if C(j,ii) <= -0.99*CC(j,ii)
            C(j,ii) = -0.99*CC(j,ii);
        elseif C(j,ii) > 0.99*(1-CC(j,ii))
            C(j,ii) = 0.99*(1-CC(j,ii));
        end
        if CN(ii) <= 0
            C(j,ii) = -0.99*CC(j,ii);
        end
    end
    if CN(icRfSO3) <= 0
        C(j,icRfSO3) = -0.99*CC(j,icRfSO3);
    end
    if CN(icRfalphaO) <= 0
        C(j,icRfalphaO) = -0.99*CC(j,icRfalphaO);
    end
    if CN(icRfbetaO) <= 0
        C(j,icRfbetaO) = -0.99*CC(j,icRfbetaO);
    end
    if CN(icH2O2) <= 0
        C(j,icH2O2) = -0.99*CC(j,icH2O2);
    end
    if CN(icHF) <= 0
        C(j,icHF) = -0.99*CC(j,icHF);
    end
    if abs(C(j,imuCe)) > 1e3
        C(j,imuCe) = 1e3*sign(C(j,imuCe));
    end
    if fCe0 > 0
        if C(j,inCe) <= -0.99*CC(j,inCe)
            C(j,inCe) = -0.99*CC(j,inCe);
        end
        if CN(inCe) <= 0
            C(j,inCe) = -0.99*CC(j,inCe);
        end
    end
end
end
```

B.2 Mechanical Model

```
function [lambda,Y,Sx,E,phip,dphip,epsw,epswz,epspq,Rs,...
    eps_hole] = plastic_model(k,C,bound,EW,R0,eps_hole,...
    A_cell,fCe0,FRR,lambda,dphip,phip,epsw,epspq,epswz,Sx,E,Y,Rs,t_vec)

    % unknowns at each mesh point
    imuw = 5; iT = 15; ipg = 16; ifCe = 19;

    % mechanical properties
    v = 0.40;      % Poisson's ratio
    Edry = 300;    % Young's modulus of dry polymer (MPa)
    Sdry = 7.5;    % yield strength of dry polymer (MPa)
    Sz = -1/Edry;  % pressure applied to the membrane (MPa)
    m = 3.6;       % scaling exponent for E
    p = 2.4;       % scaling exponent for sigmaY
    h = 2.2;       % hardening exponent
    sr = 0.33;     % Swelling (Assuming isotropic, it can be changed)
                   % a value between 0 and 1.
                   % setting 1/3 redistributes volume change
                   % (due to increase in lambda)
                   % in three directions equally
    MW = 18.016;   % molecular weight of water (g/mol)
    rho_mem = 2;   % dry membrane density (g/cm3)

    Sx0 = v/(1-v)*Sz; % intial stress
    R = 8.314;        % ideal gas constant (J/mol K)

    if k == 0 % initialize problem
        dphip = 0;  Rs = 1;   Sx = Sx0;   epspq = 0;
        for j = bound(3):bound(4)
            % density of water (g/cm3)
            rho_w(j-bound(3)+1) = 1.1603-5.371e-4*C(j,iT);
            % molar volume of water (cm3/mol)
            V0(j-bound(3)+1) = MW/rho_w(j-bound(3)+1);
            a(j-bound(3)+1) = exp((C(j,imuw)-...
                0.1*V0(j-bound(3)+1)*C(j,ipg))/(R*C(j,iT)));
            if fCe0 == 0
                b3 = 36; b2 = -42.8; b1 = 20.45; b0 = 0.05;
                lam(j-bound(3)+1) = b3*a(j-bound(3)+1)^3+...
                    b2*a(j-bound(3)+1)^2+b1*a(j-bound(3)+1)+b0;
            else
                lam(j-bound(3)+1) = 1.426+9.88*a(j-bound(3)+1)+...
                    0.1256*C(j,ifCe)-14.73*a(j-bound(3)+1)^2+...
                    2.826*a(j-bound(3)+1)*C(j,ifCe)+...
                    14.24*a(j-bound(3)+1)^3-...
                    4.0406*a(j-bound(3)+1)^2*C(j,ifCe);
            end
        end
        lambda(1) = mean(lam);
        Tavg(1) = mean(C(bound(3):bound(4),iT));
        phip(1) = 1/(1+lambda(1)*MW*rho_mem/EW);
        epsvw(1) = log(phip(1)^(-1));
        epsw(1) = epsvw(1)*sr;
        epswz(1) = epsvw(1) - 2*epsw(1);
        Ecorr = (-0.7851*sum(FRR)+136.8)/136.8; % correct for FRR
```

```
            Y(1) = 1.25*(Sdry/Edry)*phip(1)^p*Ecorr;
            E(1) = (4-0.01*Tavg(1))*phip(1)^m*Ecorr; %E/Edry
            eps_hole(1) = (pi*R0^2)/A_cell;
        else
            for j = bound(3):bound(4)
                % density of water (g/cm3)
                rho_w(j-bound(3)+1) = 1.1603-5.371e-4*C(j,iT);
                % molar volume of water (cm3/mol)
                V0(j-bound(3)+1) = MW/rho_w(j-bound(3)+1);
                a(j-bound(3)+1) = exp((C(j,imuw)-...
                    0.1*V0(j-bound(3)+1)*C(j,ipg))/(R*C(j,iT)));
                if fCe0 == 0
                    b3 = 36; b2 = -42.8; b1 = 20.45; b0 = 0.05;
                    lam(j-bound(3)+1) = b3*a(j-bound(3)+1)^3+...
                        b2*a(j-bound(3)+1)^2+b1*a(j-bound(3)+1)+b0;
                else
                    lam(j-bound(3)+1) = 1.426+9.88*a(j-bound(3)+1)+...
                        0.1256*C(j,ifCe)-14.73*a(j-bound(3)+1)^2+...
                        2.826*a(j-bound(3)+1)*C(j,ifCe)+...
                        14.24*a(j-bound(3)+1)^3-...
                        4.0406*a(j-bound(3)+1)^2*C(j,ifCe);
                end
            end
            lambda(k+1) = mean(lam);
            Tavg(k+1) = mean(C(bound(3):bound(4),iT));

            phip(k+1) = 1/(1+lambda(k+1)*MW*rho_mem/EW);
            epsvw(k+1) = log(phip(k+1)^(-1));
            epsw(k+1) = epsvw(k+1)*sr;
            epswz(k+1) = epsvw(k+1) - 2*epsw(k+1);
            Ecorr = (-0.7851*sum(FRR)+136.8)/136.8; % correct for FRR
            E(k+1) = (4-0.01*Tavg(k+1))*phip(k+1)^m*Ecorr;
            Y(k+1) = 1.25*(Sdry/Edry)*phip(k+1)^p;

            dlam(k+1)= lambda(k+1) - lambda(k);
            dphip(k+1)= phip(k+1) - phip(k);
            depsw(k+1)= epsw(k+1) - epsw(k);

            LHS = (0.5/h)/Y(k+1)+(1-v)/E(k+1);
            RHS = -depsw(k+1)*Y(k+1)/(Sx(k)-Sz)+...
                (0.5*p/h)*dphip(k)/phip(k);
            dYY(k+1) = RHS/LHS;

            if dlam(k+1) >= 0 % swelling
                dSx(k+1) = -E(k)/(1-v)*depsw(k+1);
                Sx(k+1) = Sx(k)+dSx(k+1);
                Sm(k+1) = abs(Sx(k+1)-Sz);
                if Sm(k+1) < Y(k+1) % elastic
                    Pd(k+1) = 0;
                    depsp(k+1) = 0;
                    Y(k+1) = Y(k+1);
                elseif Sm(k+1) >= Y(k+1)  % plastic
                    Pd(k+1) = 1;
                    Y(k+1) = Y(k+1)+dYY(k+1);
                    Y(k+2) = Y(k+1);
                    Sx(k+1) = -Y(k+1)+Sz;
                    dSx(k+1) = ((Sx(k+1)-Sz)/Y(k+1))*dYY(k+1);
                    depsp(k+1)= -(1-v)*E(k+1)*dSx(k+1)-depsw(k+1);
                end
            elseif dlam(k+1) < 0 % deswelling
                dSx(k+1) = -E(k)/(1-v)*depsw(k+1);
                Sx(k+1) = Sx(k)+dSx(k+1);
                Sm(k+1) = abs(Sx(k+1)-Sz);
                if Sm(k+1) < Y(k+1) % elastic
                    Pd(k+1) = 0;
                    depsp(k+1) = 0;
                    Y(k+1) = Y(k+1);
                elseif Sm(k+1) >= Y(k+1) % plastic
                    Pd(k+1) = 1;
                    Y(k+1) = Y(k+1)+dYY(k+1);
                    Y(k+2) = Y(k+1);
                    Sx(k+1) = Y(k+1)+Sz;
                    dSx(k+1) = ((Sx(k+1)-Sz)/Y(k+1))*dYY(k+1);
                    depsp(k+1)= -(1-v)*E(k)*dSx(k+1)-depsw(k+1);
                end
            end

            % mean stress
            Smean(k+1) = (2*Sx(k+1)+Sz)/3;

            % radius growth rates
            dRRs(k+1) = 0.283*abs(2*depsp(k+1))*exp(1.5*Smean(k+1)/Sm(k+1));
            Rs(k+1) = Rs(k)*(1+dRRs(k+1));

            % accumulated plastic strain
            epspq(k+1) = epspq(k)+abs(depsp(k+1));
            eps_hole(k+1) = (pi*(R0*Rs(k+1))^2)/A_cell;
        end
    end
```

B.3 Synergistic Mechanical and Chemical Degradation Model

```
% input file that specifies simulation conditions and runs the
% simulation

n = 31;           % number of unknowns at each mesh point
nregion = 5;      % number of layers in the fuel cell sandwich
nspecies = 4;     % number of gaseous species

% space (1D) discretization
njs = [40 40 40 40 40];
nj = sum(njs)+1; % total number of mesh points

% internal boundaries
bound(1) = 1;                           % anode gas channel
bound(2) = njs(1)+1;                    % anode diffusion media/
                                        % catalyst layer
bound(3) = njs(1)+njs(2)+1;             % anode catalyst layer/
                                        % membrane
bound(4) = njs(1)+njs(2)+njs(3)+1;      % membrane/
                                        % cathode catalyst layer
bound(5) = njs(1)+2*njs(2)+njs(3)+1;    % cathode catalyst layer/
                                        % diffusion media
bound(6) = nj;                          % cathode gas channel

% fuel cell dimensions
Lmem = 0.0025;  % thickness of membrane (cm)
LCLa = 0.001;   % thickness of anode catalyst layer (cm)
LCLc = 0.001;   % thickness of cathode catalyst layer (cm)
LGDL = 0.025;   % thickness of gas diffusion layers (cm)
L = [LGDL LCLa Lmem(1) LCLc LGDL]; % thickness of each layer (cm)

% simulation options
IVmode = 1;     % 1 - specify current
                % 2 - specify potential
flowmode = 1;   % 1 - constant stoichiometry
                % 2 - flow
degkin = 1;     % 1 - empirical degradation
                % 2 - microkinetic degradation

% operating conditions
iv = 0.1;       % applied current density (A/cm2) or cell potential (V)
P = 1;          % pressure (bar)
T = 80;         % temperature (deg C)
Tk = T+273.15;  % temperature (K)
RHa = 0.3;      % relative humidity at the anode
RHc = 0.3;      % relative humidity at the cathode
Pwsat = exp(11.6832-3816.44/(Tk-46.13)); % water vapor pressure (bar)
sigma = 7;      % bulk solid-phase conductivity (S/cm)
EW = 1100;      % membrane equivalent weight (g/mol)
rho_m = 2.1;    % membrane dry density (g/cm3)
lfeed = 20;     % H2 flow rate (sccm)
lair = 10;      % O2 flow rate (sccm)
Acell = 50;     % fuel cell cross sectional area (cm2)
R0 = 0.02;      % initial pinhole radius (cm)
eps_hole(1) = (pi*R0^2)/Acell; % initial pinhole void fraction
```

```
fCe0 = 0.001;  % fraction of S03- sites occupied by cerium

% kinetic parameters
a12 = 80000;   % electrode specific interfacial area (1/cm)
phimtH2 = 8e3; % thiele mass transfer for hydrogen (bar cm3 s/mol)
phimtO2 = 6e3; % thiele mass transfer for oxygen (bar cm3 s/mol)
kw = 1000;     % water vapor/membrane mass transfer coefficient
htcoeff = 1;   % heat transfer coefficient (W/cm2 K)

% transport parameters
eps0 = [0.6 0.5 eps_hole 0.5 0.6]; % void fractions for gas transport
epsM = [0.0 0.3 1-eps_hole 0.3 0.0]; % volume fraction of ionomer
% absolute permeability (cm2)
perm = [6e-8 8e-12 eps_hole*(2*R0)^2/32 8e-12 6e-8];
fwet = [0.6 0.3 0.0 0.3 0.6]; % fraction of hydrophilic pores
eta = [1.7 4.0 0.0 4.0 1.7]; % teflon loading
rad = [6 0.125 R0*10^4 0.125 6]; % characteristic pore size (um)
% effective thermal conductivity (W/cm K)
thcond = [0.015 0.003 0.0025 0.003 0.015];

itmax = 25;      % maximum number of iterations for autoband function

params = [n nregion nspecies njs bound L IVmode flowmode degkin...
    iv P Tk RHa RHc Pwsat sigma EW rho_m lfeed lair a12 phimtH2...
    phimtO2 kw htcoeff Acell eps_hole fCe0 eps0 epsM perm fwet...
    eta rad thcond];

%% run steady-state simulation
load C_ss C_ss % load initial guess
C_ss0 = C_ss;

[C_ss,iflag] = steady_state(params,itmax,C_ss0);

FRR = 0;
ploton = 1;

if eps_hole(1) > 0
    % run mechanical model
    [lambda,Y,Sx,E,phip,dphip,epsw,epswz,epspq,Rs,eps_hole] =...
        plastic_model(0,C_ss,bound,EW,R0,eps_hole,Acell,fCe0,FRR);
    % update pinhole radius
    params(40) = eps_hole(1);
    params(44) = eps_hole(1); % void fractions for gas transport (mem)
    params(49) = 1-eps_hole(1); % volume fraction of ionomer (mem)
    % absolute permeability (cm2) (mem)
    params(54) = eps_hole*(2*R0*Rs(1))^2/32;
    params(69) = R0*10^4*Rs(1); % characteristic pore size (um)
end

%% run transient simulation
dt0 = 2;      % initial time step size
dt = dt0;     % time step size
ddt = 0.5;    % delta dt for modifying time step size
dtmax = 5;    % maximum time step allowed
dtmin = 0.5; % minimum time step allowed
params = [params dt];

```

```
% set up trapezoidal cycle
n_cycle = 10;             % number of cycles
t_ramp = 20;             % time (s) of ramp portion of cycle
t_hold = 40-t_ramp;     % time (s) for steady portion of cycle
h_cycle = 0.55;           % amplitude of cycle
t_i = 6;                % initial hold time (s)
t_cycle = 2*t_ramp + 2*t_hold;  % total time for cycle (s)
tfinal = n_cycle*t_cycle + t_i; % final time (s)

t_vec = [];  % keep track of time stepping
dt_vec = []; % keep track of dt values

% initial condition
Cp = C_ss;

Ct = zeros(nj,n,[]);
Ct(:,:,1) = reshape(Cp,nj,n,1);

time = 0; % initial time (s)
t_vec = [t_vec time];

k = 0;     % number of time steps
fprintf("t = %f s\n\n",time);

tflag = 0;

while time < tfinal
    k = k+1;    % update number of time steps
    % calculates the next value in the cycle
    % for potential cycling use V_cycle function
    [RHa(k+1),RHc(k+1)] = RH_cycle(t_vec(k)+dt,dt,...
            RHa(k),RHc(k),RHa(1),t_ramp,t_hold,t_i,h_cycle);
    % update RH values
    params(26) = RHa(k+1); % relative humidity
    params(27) = RHc(k+1); % relative humidity
    % for potential cycling use
    % params(23) = V(k+1);

    time = t_vec(k)+dt; % update time step

    [C,iflag] = transient(params,itmax,Cp);

    if iflag == 1 % repeat time step if simulation did not converge
        dt = -ddt+dt;
        if dt < dtmin
            dt = dtmax;
        end
        dt_vec(k) = dt;
        k = k-1;
        C = Cp;
        fprintf("\n Simulation did not converge.\n");
    else
        dt = dt0;
        t_vec = [t_vec time];
        dt_vec = [dt_vec dt];
        fprintf("\n Simulation has converged at t = %4.1f s.\n\n",time)

        % mol -> umol
        FRR(k+1) = (abs(C(1,26))+abs(C(end,26)))*Acell*dt_vec(k)*10^6;

        if eps_hole(1) > 0
        % run mechanical model
        [lambda,Y,Sx,E,phip,dphip,epsw,epswz,epspq,Rs,eps_hole] =...
            plastic_model(k,C,bound,EW,R0,eps_hole,Acell,fCe0,...
            FRR,lambda,dphip,phip,epsw,epspq,epswz,Sx,E,Y,Rs,t_vec);
        % update pinhole radius
            params(40) = eps_hole(k+1);
            params(44) = eps_hole(k+1);
            params(49) = 1-eps_hole(k+1);
            params(54) = eps_hole(k+1)*(2*R0*Rs(k+1))^2/32;
            params(69) = R0*10^4*Rs(k+1);
        end
        % Membrane Thinning (optional)
        if sum(FRR)/Acell > 1
            Lmem(k+1) = -2.2e-4*log(sum(FRR)/A_cell) + Lmem(1);
        else
            Lmem(k+1) = Lmem(k);
        end
        params(17) = Lmem(k+1);
        Cp = C;
        Ct(:,:,k+1) = reshape(C,nj,n,1);
    end
end
```

```matlab
function [alpha00,alpha0Ce,alphaCeCe,xi,tCe,kappa] =...
    calc_cerium_props(lam,fH,T0,params)
% all units in SI
% refer to A. Crothers, et al. JES Part 1-3 (2020)

%% Physical Constants
M0 = 18.01528/1000;                % molecular weight of water (kg/mol)
F = 96485.33289;                   % Faraday's constant (C/mol)
R = 8.3144598;                     % ideal gas constant (J mol^-1 K^-1)
epsilon_0 = 8.854187817620389e-12; % vacuum permittivity (F m^-1)

%% Membrane Properties
EW = params(30); % membrane equivalent weight (g/mol)
rho_m = params(31); % dry membrane density (g/cm3)
rho0 = 1.1603-5.371e-4*T0;           % density of water (g/cm3)
V0 = M0*1000/rho0/100^3;             % molar volume of water (m3/mol)
Vp = EW/rho_m/100^3;                 % molar volume of polymer (m3/mol)
eta0 = 1e-6*(2695.3-6.6*(25+273.15)); % viscosity of water (Pa*s)

% Cerium
zCe = 3;              % valance Ce3+
D0_Ce = 0.620E-9;     % diffusion coefficient
MW_Ce = 140.91/1000;  % molecular weight (kg/mol)
Vvis_Ce = 0.154251449;

% Hydrogen
D0_H = 9.311E-9; % diffusion coefficient
MW_H = 1.0/1000; % molecular weight (kg/mol)
Vvis_H = 6.02E+23*pi/6*(4.07E-10*1)^3*1000;

D_mat = [0 D0_H D0_Ce; 0 0 Inf; 0 0 0];
D_mat = D_mat + D_mat';

%% inputs to cerium model
zs = [0, 1, zCe, -1];      % species valance, including water
zs_una = [0, 1, zCe, -1]; % macroscopic valance
Mis = [MW_H, MW_Ce];       % molecular weight of mobile, nonwater species
Ds_0 = D_mat;
Is_ref = 3;              % reference ionic strength that Dij is given at
Vvis = [Vvis_H*1.0, Vvis_Ce, 0*1]; % ion viscosity volume
d0 = 2.7E-7/100;  % m input in cm dry domain spacing
m_scaling = 1.33; % scaling of domain spacing
struct_fact = 4;  % local pore shape factor
epsr = 78.301;
R_stern = 2.751E-10; % Stern thickness of water

% calculate volume fraction of polymer
phip = 1 - lam*V0/(V0*lam + Vp); %volume fraction polymer

% calculate molalities
ms = [fH/lam/M0 (1-fH)/zCe/lam/M0 1/lam/M0] + 1E-6;  %molalities

ass_frac = eye(3);  % for no ion pairing, identity matrix

% calculate concentrations
% concentration in mole/m^3. Assuming total volume =1/V0*100^3 (m3)
Cs = [(1/V0 - sum(ms(1:2)*1000)) ms(1:2)*1000];
```

```

fCe = 1-fH;
phi0 = 1-phip;
phi_crit = 0.47*fCe+0.082-(0.43*fCe-0.016)/(1+exp(-100*fCe+1.84));
if phi0 < phi_crit
    tau = 1e6;
else
    tau = (phi0-phi_crit)^-0.95;
end

% viscofication
eta = 1e-6*(2695.3-6.6*T0)*(1 + (ms*Vvis')/2)/(1 - ms*Vvis')^2;

% calculate kR factor
Rpore = d0*phip^-m_scaling*sqrt(1-phip)/2; % pore radius (m)
if Rpore < 1e-12
    Rpore = 1e-12;
end
IDL = sqrt(((ms(1:end-1)*(zs(2:end-1).^2)')*1000*F^2)/...
    (R*T0*epsilon_0*epsr)); % inverse Debye length (1/m)
alpha_ratio = (ms(1:end-1)*zs(2:end-1)')/...
    (ms(1:end-1)*(zs(2:end-1).^2)');
if Rpore - R_stern < 1e-12
    beta = 1e-12/Rpore;
    % ratio of effective pore radius traversed by ions (cylinder)
else
    beta = (Rpore - R_stern)/Rpore;
end
kR_factor = beta^2*(2 - beta^2 - alpha_ratio*zs(1:end-1)*beta^2 -...
    (8*alpha_ratio*zs(1:end-1))/(Rpore*IDL)^2 +...
    (4*alpha_ratio*beta*zs(1:end-1)*besseli(0,beta*Rpore*IDL))/...
    (Rpore*IDL*besseli(1,beta*Rpore*IDL)) + 4*beta^2*log(1/beta) +...
    4*beta^2*log(beta)); % cylinder w/ stern V2

kR_factor(1) = 1;
% factor to account for distribution of ions throughout pores

Is = (rho0/2)*(zs(2:end).^2*ms'); % ionic strength
Ds = zeros(length(ms)); % setting up a zero matrix
Ds(1,:) =  Ds_0(1,:)*eta0/eta; % assigning solvent-zero matrix as zero
Ds(:,1) =  Ds_0(:,1)*eta0/eta;
Ds(2:end,2:end) = Ds_0(2:end,2:end)*sqrt(Is/Is_ref);
for i = 1:length(Ds)
    for j = 1:length(Ds)
        if Ds(i,j) == 0
            Ds(i,j) = Inf;
        end
    end
end

wis_ = ms(1:end-1).*Mis/(ms(1:end-1)*Mis' + 1);
wis = [1-sum(wis_) wis_];  % weight fraction of mobile species
Kij = (R*T0*(Cs'*Cs))./(Ds*sum(Cs));

Rimc = (d0*phip^-m_scaling*sqrt(1-phip))^2/(16*struct_fact*eta)*...
    (kR_factor); % Rim/ci hydrodnyamics factor, cylinder
Kim = wis./Rimc;
```

```
for j = 1:length(Cs)
    Kim = Kim + (Kij(:,j)').*(Rimc(j)./Rimc - 1);
end

Mmat_micro = zeros(length(ms));  %initializing zero matrix
for i = 1:length(ms)  % assigning the nondiagonal values
    for j = 1:length(ms)
        if j == i
            Mmat_micro(i,j) = 0;
        else
            Mmat_micro(i,j) = (R*T0*Cs(i)*Cs(j))/(Ds(i,j)*sum(Cs));
        end
    end
end
for i = 1:length(ms) % assigning the diagonals
    Mmat_micro(i,i) = -sum(Mmat_micro(i,:)) - Kim(i);
end

Mmat_macro = inv(ass_frac*inv(Mmat_micro)*(ass_frac'));

Lmat_macro = -inv(Mmat_macro)*(1 - phip)/tau;

ass_frac_upside =  1./ass_frac(2:end, 2:end);
for i = 1:length(ass_frac_upside)
    for j = 1:length(ass_frac_upside)
        if ass_frac_upside(i,j) == Inf
            ass_frac_upside(i,j) = 0;
        end
    end
end

% molality of themrodynamics species
ms_una = max(ass_frac_upside.*ms(1:end-1),[],1);
Cs = [(1/V0 - sum(ms_una(1:2)*1000)) ms_una(1:2)*1000];
kappa = F^2*sum(sum(Lmat_macro.*(zs_una(1:end-1)'*...
    zs_una(1:end-1)).*(Cs'*Cs)));

% normalized transference number ti/zi.
% t0/z0 is the electroosmotic coefficient
ti_over_zi = (Cs*F^2/kappa).*(sum(Lmat_macro.*zs_una(1:end-1).*Cs, 2))';

% alpha coefficient
alphaij = Lmat_macro.*(Cs'*Cs) - (ti_over_zi')*(ti_over_zi).*kappa/F^2;

alpha00 = alphaij(1,1);    % water-water transport
alpha0Ce = alphaij(1,3);   % water-cerium
alphaCeCe = alphaij(3,3); % cerium-cerium
xi = ti_over_zi(1);        % Electro-osmotic coefficient
tCe = 1 - ti_over_zi(2);   % transference number
end
```

```
function [Hgasin,Hgasout] = calc_enthalpy(j,Tin,C,GC)
    iyO2 = 11; iyH2 = 14; iT = 15;
    Tref = 298.15; % reference temperature (K)
    % Define constant physical properties
    % Component      O2         N2          H2O(v)      H2
    a = [28.11         31.15       32.24      27.14    ];
    b = [-3.68e-6     -1.357e-2    1.924e-3   9.274e-3 ];
    c = [1.746e-5      2.68e-5     1.055e-5   -1.381e-5];
    d = [-1.065e-8    -1.168e-8   -3.596e-9   7.645e-9 ];
    Hin = a.*(Tin-Tref)+(b./2).*(Tin^2-Tref^2)+...
        (c./3).*(Tin^3-Tref^3)+(d./4).*(Tin^4-Tref^4);
    Hout = a.*(C(j,iT)-Tref)+(b./2).*(C(j,iT)^2-Tref^2)+...
        (c./3).*(C(j,iT)^3-Tref^3)+(d./4).*(C(j,iT)^4-Tref^4);
    Hgasin = Hin*GC;
    Hgasout = Hout*C(j,iyO2:iyH2)';
end
```

```
function [f,lambda,C0V] = calc_lambda(j,C,params)
imuw = 5; iT = 15; ipg = 16; ifCe = 19;

EW = params(30); % membrane equivalent weight (g/mol)
rho_m = params(31); % dry membrane density (g/cm3)
MW = 18.016;       % molecular weight of water (g/mol)
rho_w = 1.1603-5.371e-4*C(j,iT); % density of water (g/cm3)
V0 = MW/rho_w;     % molar volume of water (cm3/mol)
Vm = EW/rho_m;     % molar volume of the membrane (cm3/mol)
R = 8.314;         % ideal gas constant (J/mol K)
fCe0 = params(41);

a = exp((C(j,imuw)-0.1*V0*C(j,ipg))/(R*C(j,iT)));
if fCe0 == 0
    b3 = 36; b2 = -42.8; b1 = 20.45; b0 = 0.05;
    lambda = b3*a^3+b2*a^2+b1*a+b0;
else
    lambda = 1.426+9.88*a+0.1256*C(j,ifCe)-14.73*a^2+...
        2.826*a*C(j,ifCe)+14.24*a^3-4.0406*a^2*C(j,ifCe);
end
f = (lambda*V0)/(Vm+lambda*V0); % water volume fraction in membrane
C0V = lambda/(Vm+lambda*V0);
end
```

```
function Lmem = calc_Lmem(mode,j,C,params)
iT = 15;

EW = params(30); % membrane equivalent weight (g/mol)
rho_m = params(31); % dry membrane density (g/cm3)
MW = 18.016;       % molecular weight of water (g/mol)
rho_w1 = 1.1603-5.371e-4*C(j,iT); % density of water (g/cm3)
V01 = MW/rho_w1;     % molar volume of water (cm3/mol)
Vm = EW/rho_m;     % molar volume of the membrane (cm3/mol)

[~,lambda1] = calc_lambda(j,C,params);
Lmem1 = 1+0.36*lambda1*V01/Vm;
if mode == 1
    rho_w2 = 1.1603-5.371e-4*C(j-1,iT); % density of water (g/cm3)
    V02 = MW/rho_w2;     % molar volume of water (cm3/mol)
    [~,lambda2] = calc_lambda(j-1,C,params);
    Lmem2 = 1+0.36*lambda2*V02/Vm;
else
    rho_w2 = 1.1603-5.371e-4*C(j+1,iT); % density of water (g/cm3)
    V02 = MW/rho_w2;     % molar volume of water (cm3/mol)
    [~,lambda2] = calc_lambda(j+1,C,params);
    Lmem2 = 1+0.36*lambda2*V02/Vm;
end
Lmem = 0.5*(Lmem1+Lmem2);
end
```

```
function P = calc_pressure(mode,iregion,j,C,params)
iNO2 = 7; iNN2 = 8; iNw = 9; iNH2 = 10;
iyO2 = 11; iyH2 = 14; iT = 15; ipg = 16;

MW = [31.9988; 28.014; 18.0152; 2.0159]; % molecular weight (g/mol)
perm = params(52:56);
R = 83.14;  % ideal gas constant (cm3 bar/mol K)
dx = mesh(j,C,iregion,params);
if mode == 1
    CT = (C(j,ipg)/C(j,iT)+C(j-1,ipg)/C(j-1,iT))/2/R;
    gasmass = ((C(j-1,iyO2:iyH2)+C(j,iyO2:iyH2))/2)*MW;
    gasflux = [C(j-1,iNO2:iNN2) C(j,iNw:iNH2)]*MW;
    gasvel = 1/CT*gasflux/gasmass;
    visc = viscgas(C(j,iT),C(j,iyO2:iyH2));
    viscL = viscgas(C(j-1,iT),C(j-1,iyO2:iyH2));
    visgL = (visc+viscL)/2;
    P = gasvel+perm(iregion)/visgL*(C(j,ipg)-C(j-1,ipg))/dx;
elseif mode == 2
    CT = (C(j,ipg)/C(j,iT)+C(j+1,ipg)/C(j+1,iT))/2/R;
    gasmass = ((C(j,iyO2:iyH2)+C(j+1,iyO2:iyH2))/2)*MW;
    gasflux = [C(j,iNO2) C(j,iNN2) C(j,iNw) C(j+1,iNH2)]*MW;
    gasvel = 1/CT*gasflux/gasmass;
    visc = viscgas(C(j,iT),C(j,iyO2:iyH2));
    viscR = viscgas(C(j+1,iT),C(j+1,iyO2:iyH2));
    visgR = (visc+viscR)/2;
    P = gasvel+perm(iregion)/visgR*(C(j+1,ipg)-C(j,ipg))/dx;
end
end
```

```matlab
function [kappa,xi,alphaCe0,tCe,alpha00,alphaCeCe] =...
    calc_mem_props(mode,j,C,params)

iT = 15; ifCe = 19;
epsM = params(47:51); % volume fraction of ionomer
fCe0 = params(41);
bound = params(9:14);
iregion = region(mode,j,bound);

if fCe0 == 0
    [kappa1,xi1,alpha001] = calc_props(iregion,j,C,params);
    if mode == 1
        [kappa2,xi2,alpha002] = calc_props(iregion,j-1,C,params);
    else
        [kappa2,xi2,alpha002] = calc_props(iregion,j+1,C,params);
    end
    kappa = 0.5*(kappa1+kappa2);
    xi = 0.5*(xi1+xi2);
    alpha00 = 0.5*(alpha001+alpha002);
    alphaCe0 = 0;
    alphaCeCe = 0;
    tCe = 0;
else
    % the 0.1 converts cm3 bar/mol K ---> J/mol K
    [~,lambda1] = calc_lambda(j,C,params);
    [alpha001,alphaCe01,alphaCeCe1,xi1,tCe1,kappa1] = ...
        calc_cerium_props(lambda1,1-C(j,ifCe),C(j,iT),params);

    if mode == 1
        [~,lambda2] = calc_lambda(j-1,C,params);
        [alpha002,alphaCe02,alphaCeCe2,xi2,tCe2,kappa2] = ...
            calc_cerium_props(lambda2,1-C(j-1,ifCe),C(j-1,iT),params);
    else
       [~,lambda2] = calc_lambda(j+1,C,params);
        [alpha002,alphaCe02,alphaCeCe2,xi2,tCe2,kappa2] = ...
            calc_cerium_props(lambda2,1-C(j+1,ifCe),C(j+1,iT),params);
    end
    % divide 1/100 convert from S/m to S/cm
    kappa = (1/100)*0.5*(kappa1+kappa2)*epsM(iregion)^1.5;
    xi = 0.5*(xi1+xi2);
    tCe = 0.5*(tCe1+tCe2)*epsM(iregion)^1.5;
    alpha00 = (1/100)*0.5*(alpha001+alpha002)*epsM(iregion)^1.5;
    alphaCe0 = (1/100)*0.5*(alphaCe01+alphaCe02)*epsM(iregion)^1.5;
    alphaCeCe = (1/100)*0.5*(alphaCeCe1+alphaCeCe2)*epsM(iregion)^1.5;
end
end
```

```
function [condmemv,alphav,xiv] = calc_props(iregion,j,C,params)
imuw = 5; iT = 15; ipg = 16;

EW = params(30); % membrane equivalent weight (g/mol)
rho_m = params(31); % dry membrane density (g/cm3)
MW0 = 18.0152; % molecular weight of water (g/mol)
R = 8.314; % ideal gas constant (J/mol K)
Tref = 30+273.15; % reference temperature (K)
fperc = 0.06; % conductivity percolation threshold
epsM = params(47:51); % volume fraction of ionomer

% membrane properties
rho_w = 1.1603-5.371e-4*C(j,iT); % density of water (g/cm3)
V0 = MW0/rho_w;
Vm = EW/rho_m;     % molar volume of the membrane (cm3/mol)
a = exp((C(j,imuw)-0.1*V0*C(j,ipg))/(R*C(j,iT)));

% calculate water content from isotherm (Weber & Newman 2004)
b3 = 36; b2 = -42.8; b1 = 20.45; b0 = 0.05;
xlamv = b3*a^3+b2*a^2+b1*a+b0;

% calculated membrane properties
cwaterv = xlamv/(V0*xlamv+Vm);
cmemv = 1/(V0*xlamv+Vm);
fwaterv = xlamv*V0/(xlamv*V0+Vm);

% electroosmotic coefficient (mol H2O/mol H+)
if xlamv < 1
    xiv = xlamv;
else
    xiv = 1;
end
DH2Om = 1.8e-5*exp(20000/R*(1/Tref-1/C(j,iT)));
DH2O = DH2Om*fwaterv;
xwaterv = cwaterv/(cwaterv+cmemv);
alphav = cwaterv*DH2O/R/C(j,iT)/(1-xwaterv);
% membrane conductivity (S/cm)
if fwaterv < fperc
    condmemv = 1e-5;
    disp('failure: membrane conductivity is zero')
elseif fwaterv >= 0.45
    sigp = 0.5*(0.45-fperc)^1.5;
    sigmxv = exp(15000/R*(1/Tref-1/C(j,iT)));
    condmemv = sigp*sigmxv;
else
    sigp = 0.5*(fwaterv-fperc)^1.5;
    sigmxv = exp(15000/R*(1/Tref-1/C(j,iT)));
    condmemv = sigp*sigmxv;
end
% effective properties
condmemv = condmemv*epsM(iregion)^1.5;
alphav = alphav*epsM(iregion)^1.5;
xiv = xiv*epsM(iregion)^1.5;
end
```

```
function psiH2 = calc_psiH2(mode,j,C,params)
icRfSO3 = 25;  iT = 15;
R = 8.314;          % ideal gas constant (J/mol K)
Tref = 30+273.15; % reference temperature (K)
EW = params(30); % membrane equivalent weight (g/mol)
rho_m = params(31); % dry membrane density (g/cm3)
degkin = params(22);
f1 = calc_lambda(j,C,params);

% H2 permeation coefficient (mol/bar/cm/s)
psiH21 = (2.2e-11*f1+2.9e-12)*exp((21000/R)*(1/Tref - 1/C(j,iT)));
if mode == 1
    f2 = calc_lambda(j-1,C,params);
    psiH22 = (2.2e-11*f2+2.9e-12)*exp((21000/R)*(1/Tref - 1/C(j-1,iT)));
else
    f2 = calc_lambda(j+1,C,params);
    psiH22 = (2.2e-11*f2+2.9e-12)*exp((21000/R)*(1/Tref - 1/C(j+1,iT)));
end

if degkin == 1
    psiH2 =  0.5*(psiH21+psiH22);
else
    x = (C(j,icRfSO3))/(rho_m/EW);
    psiH2 =  0.5*(102*x^2-201*x+100)*(psiH21+psiH22);
end
end
```

```
function psiO2 = calc_psiO2(mode,j,C,params)
icRfSO3 = 25; iT = 15;
R = 8.314;          % ideal gas constant (J/mol K)
Tref = 30+273.15; % reference temperature (K)
EW = params(30); % membrane equivalent weight (g/mol)
rho_m = params(31); % dry membrane density (g/cm3)
degkin = params(22);
f1 = calc_lambda(j,C,params);

% O2 permeation coefficient (mol/bar/cm/s)
psiO21 = (1.9e-11*f1+1.1e-12)*exp((22000/R)*(1/Tref - 1/C(j,iT)));
if mode == 1
    f2 = calc_lambda(j-1,C,params);
    psiO22 = (1.9e-11*f2+1.1e-12)*exp((22000/R)*(1/Tref - 1/C(j-1,iT)));
else
    f2 = calc_lambda(j+1,C,params);
    psiO22 = (1.9e-11*f2+1.1e-12)*exp((22000/R)*(1/Tref - 1/C(j+1,iT)));
end

if degkin == 1
    psiO2 =  0.5*(psiO21+psiO22);
elseif degkin == 2
    x = (C(j,icRfSO3))/(rho_m/EW);
    psiO2 =  0.5*(102*x^2-201*x+100)*(psiO21+psiO22);
end
end
```

```matlab
function sigmaeff = calc_sigma(iregion,params)
sigma = params(29); % bulk-phase conductivity (S/cm)
eps0 = params(42:46); % void fractions for gas transport
epsM = params(47:51); % volume fraction of ionomer
fwet = params(57:61); % fraction of hydrophilic pores
eta = params(62:66); % teflon loading
epssolid = 1-eps0(iregion)-epsM(iregion);
eps1 = fwet(iregion)*eta(iregion)*epssolid;
sigmaeff = sigma*eps1^1.5;
end
```

```matlab
function D = diffcoeff(p,T)
% Calculates diffusion coefficients for binary gas systems
% at low pressures using kinetic theory
% see Properties of Gases and Liquids, 5th ed.
% Poling, Prausnitz, O'Connell
% pg 11.5-11.7

% gas phase species
% 1 = oxygen, 2 = nitrogen, 3 = water, 4 = hydrogen

MW = [31.9988 28.014 18.015 2.0159]; % molecular weight
sig = [3.467 3.798 2.641 2.827];     % characteristic length (Angstrom)
e = [106.7 71.4 809.1 59.7];         % Lennard-Jones energy
                                     % divided by Boltzmann's constant
                                     % k = 1.3806e-23 (m2 kg/s2 K)
sigij = 0.5*(sig+sig');
eij = (e.*e').^0.5;
MWij = 2*((1./MW)+(1./MW)').^-1;
Tstar = T./eij;
P = p/1.01325; % convert pressure from bar to atm
OmegaD = 1.06036./(Tstar.^0.15610)+0.19300./exp(0.47635.*Tstar)+...
    1.03587./exp(1.52996.*Tstar)+1.76474./exp(3.89411.*Tstar);
D = (0.00266*T^1.5)./(P*MWij.^0.5.*sigij.^2.*OmegaD); % cm2/s
end
```

```matlab
function Dk = knudsen(T,MW,rad)

R = 8.3143; % ideal gas constant (J/mol K)
MW = MW*1e-3; % molecular weight of water kg/mol
dia = 2.*rad./1e6; % mean pore radius (convert from micron to m)

% Measurements of Pore Size Distribution, Porosity, Effective Oxygen
% Diffusivity, and Tortuousity of PEM Fuel Cell Electrodes
% Z. Yu, R. N. Carter, J. Zhang
% Fuel Cells 12, 2012 No. 4, 557-565
Dk = (dia./3).*sqrt((8*R*T)./(pi*MW));
Dk = Dk*1e4; % convert from m2/s to cm2/s
end
```

```
function energyfluxleft = energyfluxleft(j,C,Cp,sigma,kappa,params)
% Calculates the flux exiting the box to the left of point j
iNwmem = 6; iNO2 = 7; iNN2 = 8; iNw = 9; iNH2 = 10;
iyO2 = 11; iyH2 = 14; iT = 15; ipg = 16;

% parameters
bound = params(9:14);
nj = bound(6);
F = 96485; % Faraday's constant (C/mol)
R = 83.14; % ideal gas constant (cm3 bar/mol K)

iregion = region(1,j,bound);
dx = mesh(j,C,iregion,params);

% effective thermal conductity (W/cm K)
thcond = params(72:76);
% Conduction
cond = -thcond(iregion)*(C(j,iT)-C(j-1,iT))/dx;
if j ~= 1
    a = [28.11         31.15       32.24      27.14    ];
    b = [-3.68e-6     -1.357e-2    1.924e-3   9.274e-3 ];
    c = [1.746e-5      2.68e-5     1.055e-5   -1.381e-5];
    d = [-1.065e-8    -1.168e-8   -3.596e-9   7.645e-9 ];
    Cpr = a + b.*C(j-1:j,iT) + c.*(C(j-1:j,iT)).^2 +d.*(C(j-1:j,iT)).^3;
    Cpgas = Cpr*C(j-1:j,iyO2:iyH2)';
    CpgasL = (Cpgas(1,1)+Cpgas(2,2))/2;
    if j > bound(2) && j <= bound(5)
        Cpw = (2.7637e5-2090.1*C(j-1:j,iT)+8.125*C(j-1:j,iT).^2-...
            1.4116e-2*C(j-1:j,iT).^3+9.3701e-6*C(j-1:j,iT).^4)/1000;
        CpwL = 0.5*(Cpw(1)+Cpw(2));
    end
end

% Convection (Flux into the box to the left)
if j > 1 && j <= bound(3)
    gasflux = C(j,iNH2)+C(j,iNw)+C(j-1,iNO2)+C(j-1,iNN2);
elseif j > bound(4) && j <= nj
    gasflux = C(j-1,iNO2)+C(j-1,iNN2)+C(j-1,iNw)+C(j,iNH2);
else
    gasflux = 0;
end

if j == 1
    conv = 0;
elseif j > 1 && j <= bound(2)
    conv = CpgasL*gasflux*((C(j,iT)+C(j-1,iT))/2);
elseif j > bound(2) && j <= bound(3)
    conv = (CpgasL*gasflux+CpwL*C(j-1,iNwmem))*((C(j,iT)+C(j-1,iT))/2);
elseif j > bound(3) && j <= bound(4)
    conv = CpwL*C(j-1,iNwmem)*((C(j,iT)+C(j-1,iT))/2);
elseif j > bound(4) && j <= bound(5)
    conv = (CpgasL*gasflux+CpwL*C(j-1,iNwmem))*((C(j,iT)+C(j-1,iT))/2);
elseif j > bound(5) && j < nj
    conv = CpgasL*gasflux*((C(j,iT)+C(j-1,iT))/2);
elseif j == nj
    conv = 0;
end
```

```

% Reaction terms
% reaction 1 = HOR heat generation
% reaction 2 = ORR heat generation
% reaction 3 = ohmic heating
st = [1 2*F -4*F];

heat = heat_react(j,iregion,C,sigma,kappa,params);
if j ~= 1
    heatL = heat_react(j-1,iregion,C,sigma,kappa,params);
else
    heatL = heat;
end
w = 0.75;
gen = st*(w*heat+(1-w)*heatL)*dx/2;

acc = 0;
if isempty(Cp) == 0 % transient
    dt = params(77); % time spacing
    a = [28.11         31.15       32.24      27.14    ];
    b = [-3.68e-6     -1.357e-2    1.924e-3   9.274e-3 ];
    c = [1.746e-5      2.68e-5     1.055e-5   -1.381e-5];
    d = [-1.065e-8    -1.168e-8   -3.596e-9   7.645e-9 ];
    Cprp = a + b.*Cp(j-1:j,iT) + c.*(Cp(j-1:j,iT)).^2 +...
        d.*(Cp(j-1:j,iT)).^3;
    Cpgasp = Cp(j-1:j,iyO2:iyH2)*Cprp';
    CpgasLp = (Cpgasp(1,1)+Cpgasp(2,2))/2;
    CT = (C(j,ipg)/C(j,iT)+C(j-1,ipg)/C(j-1,iT))/(2*R);
    CTp = (Cp(j,ipg)/Cp(j,iT)+Cp(j-1,ipg)/Cp(j-1,iT))/(2*R);
    dTdt = w*0.5*(CpgasL*CT*C(j,iT)-CpgasLp*CTp*Cp(j,iT))/dt+...
         (1-w)*0.5*(CpgasL*CT*C(j-1,iT)-CpgasLp*CTp*Cp(j-1,iT))/dt;
    acc = dTdt*dx/2;
end

% Flux leaving the box to the left
energyfluxleft = cond + conv + gen - acc;
end
```

```
function energyfluxright = energyfluxright(j,C,Cp,sigma,kappa,params)
% Calculates the flux exiting the box to the left of point j
iNwmem = 6; iNO2 = 7; iNN2 = 8; iNw = 9; iNH2 = 10;
iyO2 = 11; iyH2 = 14; iT = 15; ipg = 16;

% parameters
bound = params(9:14);
nj = bound(6);
F = 96485; % Faraday's constant (C/mol)
R = 83.14; % ideal gas constant (cm3 bar/mol K)

iregion = region(2,j,bound);
dx = mesh(j,C,iregion,params);

% effective thermal conductity (W/cm K)
thcond = params(72:76);
% Conduction
cond = -thcond(iregion)*(C(j+1,iT)-C(j,iT))/dx;
if j ~= nj
    a = [28.11         31.15       32.24      27.14    ];
    b = [-3.68e-6     -1.357e-2    1.924e-3   9.274e-3 ];
    c = [1.746e-5      2.68e-5     1.055e-5   -1.381e-5];
    d = [-1.065e-8    -1.168e-8   -3.596e-9   7.645e-9 ];
    Cpr = a + b.*C(j:j+1,iT) + c.*C(j:j+1,iT).^2 +d.*C(j:j+1,iT).^3;
    Cpgas = Cpr*C(j:j+1,iyO2:iyH2)';
    CpgasR = (Cpgas(1,1)+Cpgas(2,2))/2;
    if j > bound(2) && j <= bound(5)
        Cpw = (2.7637e5-2090.1*C(j:j+1,iT)+8.125*C(j:j+1,iT).^2-...
            1.4116e-2*C(j:j+1,iT).^3+9.3701e-6*C(j:j+1,iT).^4)/1000;
        CpwR = 0.5*(Cpw(1)+Cpw(2));
    end
end

% Convection (Flux into the box to the left)
if j >= 1 && j < bound(3)
    gasflux = C(j+1,iNH2)+C(j+1,iNw)+C(j,iNO2)+C(j,iNN2);
elseif j >= bound(4) && j < nj
    gasflux = C(j,iNO2)+C(j,iNN2)+C(j,iNw)+C(j+1,iNH2);
else
    gasflux = 0;
end

if j == 1
    conv = 0;
elseif j > 1 && j <= bound(2)
    conv = CpgasR*gasflux*C(j,iT);
elseif j > bound(2) && j < bound(3)
    conv = (CpgasR*gasflux+CpwR*C(j,iNwmem))*(C(j+1,iT)+C(j,iT))/2;
elseif j >= bound(3) && j < bound(4)
    conv = CpwR*C(j,iNwmem)*(C(j+1,iT)+C(j,iT))/2;
elseif j >= bound(4) && j < bound(5)
    conv = (CpgasR*gasflux+CpwR*C(j,iNwmem))*(C(j+1,iT)+C(j,iT))/2;
elseif j >= bound(5) && j < nj
    conv = CpgasR*gasflux*(C(j+1,iT)+C(j,iT))/2;
elseif j == nj
    conv = 0;
end
```

```

% Reaction terms
% reaction 1 = HOR heat generation
% reaction 2 = ORR heat generation
% reaction 3 = ohmic heating
st = [1 2*F -4*F];
heat = heat_react(j,iregion,C,sigma,kappa,params);
if j ~= 1
    heatR = heat_react(j+1,iregion,C,sigma,kappa,params);
else
    heatR = heat;
end
w = 0.75;
gen = st*(w*heat+(1-w)*heatR)*dx/2;

acc = 0;
if isempty(Cp) == 0 % transient
    dt = params(77); % time spacing
    a = [28.11          31.15        32.24       27.14     ];
    b = [-3.68e-6     -1.357e-2     1.924e-3    9.274e-3 ];
    c = [1.746e-5      2.68e-5      1.055e-5    -1.381e-5];
    d = [-1.065e-8    -1.168e-8    -3.596e-9    7.645e-9 ];
    CT = (C(j,ipg)/C(j,iT)+C(j+1,ipg)/C(j+1,iT))/(2*R);
    CTp = (Cp(j,ipg)/Cp(j,iT)+Cp(j+1,ipg)/Cp(j+1,iT))/(2*R);
    Cprp = a + b.*Cp(j:j+1,iT) + c.*(Cp(j:j+1,iT)).^2 +...
        d.*(Cp(j:j+1,iT)).^3;
    Cpgasp = Cp(j:j+1,iyO2:iyH2)*Cprp';
    CpgasRp = (Cpgasp(1,1)+Cpgasp(2,2))/2;
    dTdt = w*0.5*(CT*CpgasR*C(j,iT)-CTp*CpgasRp*Cp(j,iT))/dt + ...
        (1-w)*0.5*(CT*CpgasR*C(j+1,iT)-CTp*CpgasRp*Cp(j+1,iT))/dt;
    acc = dTdt*dx/2;
end

% Flux leaving the box to the left
energyfluxright = cond + conv - gen + acc;

end
```

```
function eq = eqn(j,jp,k,dC,C,Cp,params)
% This function contains the governing differential equations.
%
% j = current mesh point
% eq = array of equation residuals
% jp = perturbed value of j (for numerical derivatives)
% k = identifies the perturbed variable
% dC = size of perturbation
%
% n = number of variables
% nj = number of mesh points

C(jp,k) = C(jp,k)+dC;

% unknowns at each mesh point
ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; imuw = 5; iNwmem = 6; iNO2 = 7;
iNN2 = 8; iNw = 9; iNH2 = 10; iyO2 = 11; iyN2 = 12; iyw = 13; iyH2 = 14;
iT = 15; ipg = 16; iL = 17; itau = 18; ifCe = 19; imuCe = 20; iNCe = 21;
inCe = 22; iNH2O2 = 23; icH2O2 = 24; icRfSO3 = 25; iNHF = 26; icHF = 27;
icCO2H = 28; icRfalphaO = 29; icRfbetaO = 30; icOH = 31;

% discretization
bound = params(9:14); nj = bound(6);
L = params(15:19);

%% Physical Properties
zCe = 3; % charge of cerium ion, assume Cerium is in Ce3+ form
zH = 1;  % charge of a proton
F = 96485; % Faraday's constant (C/mol)
R = 8.314; % ideal gas constant (J/mol K)
iregion1 = region(1,j,bound);
iregion2 = region(2,j,bound);
P0 = params(24); % pressure (bar)
EW = params(30); % membrane equivalent weight (g/mol)
rho_m = params(31); % dry membrane density (g/cm3)
eps_hole = params(40); % pinhole volume fraction
eps0 = params(42:46); % void fractions for gas transport
epsM = params(47:51); % volume fraction of ionomer
fCe0 = params(41); % fraction of SO3- site occupied by Ce
degkin = params(22); % empirical (1) or microkinetics (2)

%% Gas Channel Mass Balances
if j == 1 % anode gas channel
    % feed gas (can include N2 as inert), if yH2 = 1, then pure H2 feed
    yH2 = 1;
    % mole fractions in
    RHa = params(26);       % relative humidity
    Pwsat = params(28);     % water vapor pressure (bar)
    aywin = RHa*Pwsat/P0;
    ayH2in = (1-aywin)*yH2;
    ayN2in = (1-aywin)*(1-yH2);
    % total dry gas flow in
    lfeed = params(32); % feed stoich/flow
    flowmode = params(21);
    if flowmode == 1 % constant stoich
        adgNin = (C(j,ii1)/(2*F))*(lfeed/yH2);
    elseif flowmode == 2 % constant flow
```

```
        adgNin = lfeed*7.45e-7; % sccm -> mol/s
    end
    % gas flows in
    aNwin = adgNin*RHa*Pwsat/(P0-RHa*Pwsat);
    aNN2in = adgNin*(1-yH2);
    % total gas flow in
    agasNin = aNwin+adgNin;
    % total dry gas flow out
    adgNout = adgNin - C(j,iNH2) - C(j,iNN2);
    % mole fractions and gas flows out
    aywout = C(j,iyw);
    aNwout = adgNout*aywout/(1-aywout);
    aNN2out = aNN2in - C(j,iNN2);
    ayN2out = aNN2out/(aNwout+adgNout);
    aNwDM = fluxright(j,iNw,C,Cp,params);
    % total gas flow out
    agasNout = aNwout+adgNout;
elseif j == nj % cathode gas channel
    % feed gas (air)
    yO2 = 0.21;
    yN2 = 0.79;
    % mole fractions in
    RHc = params(27);       % relative humidity
    Pwsat = params(28);     % water vapor pressure (bar)
    cywin = RHc*Pwsat/P0;
    cyO2in = (1-cywin)*yO2;
    cyN2in = (1-cywin)*yN2;
    % total dry gas flow in
    lair = params(33); % air stoich/flow
    flowmode = params(21);
    if flowmode == 1 % constant stoich
        cdgNin = (C(j,ii1)/(4*F))*(lair/cyO2in);
    elseif flowmode == 2 % constant flow
        cdgNin = lair*7.45e-7; % sccm -> mol/s
    end
    % gas flows in
    cNwin = cdgNin*RHc*Pwsat/(P0-RHc*Pwsat);
    cNN2in = cdgNin*yN2;
    % total gas flow in
    cgasNin = cNwin+cdgNin;
    % total dry gas flow out
    cdgNout = cdgNin+C(j,iNO2)+C(j,iNN2);
    % mole fractions and gas flows out
    cywout = C(j,iyw);
    cNwout = cdgNout*cywout/(1-cywout);
    cNN2out = cNN2in+C(j,iNN2);
    cyN2out = cNN2out/(cNwout+cdgNout);
    cNwDM = fluxleft(j,iNw,C,Cp,params);
    % total gas flow out
    cgasNout = cNwout+cdgNout;
end

%% Governing Equations

% Equation 1: solid phase current
    if j <= bound(3)
        eq(ii1) = (C(j+1,ii2)-C(j,ii2))+(C(j+1,ii1)-C(j,ii1));
    elseif j > bound(3) && j < bound(4)
        eq(ii1) = C(j,ii1);
    else
        eq(ii1) = (C(j,ii1)-C(j-1,ii1))+(C(j,ii2)-C(j-1,ii2));
    end

% Equation 2: solid phase potential
    if j == 1
        eq(iv1) = C(j,iv1);
    elseif j > 1 && j <= bound(3)
        dx = mesh(j,C,iregion1,params);
        sigma = 0.5*(calc_sigma(iregion1,params)+...
            calc_sigma(iregion2,params));
        eq(iv1) = C(j,ii1) + sigma*(C(j,iv1)-C(j-1,iv1))/dx;
    elseif j > bound(3) && j < bound(4)
        eq(iv1) = C(j,iv1);
    elseif j >= bound(4) && j < nj
        dx = mesh(j,C,iregion2,params);
        sigma = 0.5*(calc_sigma(iregion1,params)+...
            calc_sigma(iregion2,params));
        eq(iv1) = C(j,ii1) + sigma*(C(j+1,iv1)-C(j,iv1))/dx;
    elseif j == nj
        iv = params(23);
        IVmode = params(20);
        if IVmode == 1
            eq(iv1) = C(j,ii1) - iv; % specify current density (A/cm2)
        elseif IVmode == 2
            eq(iv1) = C(j,iv1) - iv; % specify cell potential (V)
        end
    end

% Equation 3: membrane phase current
    if j < bound(2)
        eq(ii2) = C(j,ii2);
    elseif j == bound(2)
        eq(ii2) = fluxright(j,ii2,C,Cp,params);
    elseif j > bound(2) && j <= bound(5)
        eq(ii2) = fluxleft(j,ii2,C,Cp,params)-...
            fluxright(j,ii2,C,Cp,params);
    else
        eq(ii2) = C(j,ii2);
    end

% Equation 4: membrane phase potential
    if j < bound(2)
        eq(iv2) = C(j,iv2);
    elseif j >= bound(2) && j < bound(5)
        dx = mesh(j,C,iregion2,params);
        [kappa,xi,alphaCe0,tCe,~,alphaCeCe] = ...
            calc_mem_props(2,j,C,params);
        if fCe0 == 0
            eq(iv2) = C(j,ii2)+kappa*(C(j+1,iv2)-C(j,iv2))/dx +...
                ((kappa*xi)/F)*(C(j+1,imuw)-C(j,imuw))/dx;
        else
            alphaH0 = -alphaCe0*zCe;
            alphaHCe = -alphaCeCe*zCe;
            NH = -(kappa*(1-tCe)/(zH*F))*(C(j+1,iv2)-C(j,iv2))/dx -...
```

```
                (alphaH0+(kappa*xi*(1-tCe))/(zH*F^2))*...
                (C(j+1,imuw)-C(j,imuw))/dx -...
                (alphaHCe+((1-tCe)*tCe*kappa)/(zH*zCe*F^2))*...
                (C(j+1,imuCe)-C(j,imuCe))/dx;
            eq(iv2) = C(j,ii2) - (zH*NH+zCe*C(j,iNCe))*F;
        end
    elseif j == bound(5)
        eq(iv2) = fluxleft(j,ii2,C,Cp,params);
    else
        eq(iv2) = C(j,iv2);
    end

% Equation 5: Water chemical potential in the membrane
    if j < bound(2)
        eq(imuw) = C(j,imuw);
    elseif j >= bound(2) && j < bound(5)
        dx = mesh(j,C,iregion2,params);
        [kappa,xi,alphaCe0,tCe,alpha00] = ...
            calc_mem_props(2,j,C,params);
        if fCe0 == 0
            eq(imuw) = C(j,iNwmem) +....
                ((kappa*xi)/F)*(C(j+1,iv2)-C(j,iv2))/dx +...
                (alpha00+(kappa*xi^2)/F^2)*(C(j+1,imuw)-C(j,imuw))/dx;
        else
            eq(imuw)=C(j,iNwmem)+(alphaCe0+(xi*tCe*kappa)/(zCe*F^2))*...
                (C(j+1,imuCe)-C(j,imuCe))/dx + ((kappa*xi)/F)*...
                (C(j+1,iv2)-C(j,iv2))/dx + alpha00+(kappa*xi^2)/F^2)*...
                (C(j+1,imuw)-C(j,imuw))/dx;
        end
    elseif j == bound(5)
        eq(imuw) = fluxleft(j,iNwmem,C,Cp,params);
    else
        eq(imuw) = C(j,imuw);
    end

% Equation 6: Water flux in the membrane
    if j < bound(2)
        eq(iNwmem) = C(j,iNwmem);
    elseif j == bound(2)
        eq(iNwmem) = fluxright(j,iNwmem,C,Cp,params);
    elseif j > bound(2) && j <= bound(5)
        eq(iNwmem) = fluxleft(j,iNwmem,C,Cp,params) -...
            fluxright(j,iNwmem,C,Cp,params);
    else
        eq(iNwmem) = C(j,iNwmem);
    end

% Equation 7: Oxygen flux
    if j == 1
        eq(iNO2) = C(j,iNO2);
    else
        eq(iNO2) = fluxleft(j,iNO2,C,Cp,params)-...
            fluxright(j,iNO2,C,Cp,params);
    end

% Equation 8: Nitrogen flux
    if j == 1
```

```
        ayN2avg = (ayN2in+ayN2out)/2;
        eq(iNN2) = C(j,iyN2) - ayN2avg; % gas channel mass balance
    else
        eq(iNN2) = fluxleft(j,iNN2,C,Cp,params)-...
            fluxright(j,iNN2,C,Cp,params);
    end

%% NO PINHOLE
if eps_hole == 0
    % Equation 9: Water flux
    if j == 1
        eq(iNw) = aNwin - aNwDM - aNwout; % gas channel mass balance
    elseif j > 1 && j <= bound(3)
        eq(iNw) = fluxleft(j,iNw,C,Cp,params)-...
            fluxright(j,iNw,C,Cp,params);
    elseif j > bound(3) && j < bound(4)
        eq(iNw) = C(j,iNw);
    elseif j >= bound(4) && j < nj
        eq(iNw) = fluxleft(j,iNw,C,Cp,params)-...
            fluxright(j,iNw,C,Cp,params);
    elseif j == nj
        eq(iNw) = cNwin + cNwDM - cNwout; % gas channel mass balance
    end
    % Equation 10: Hydrogen flux
    if j == 1
        eq(iNH2) = C(j,iNH2) - fluxright(j,iNH2,C,Cp,params);
    elseif j > 1 && j < bound(5)
        eq(iNH2) = fluxleft(j,iNH2,C,Cp,params)-...
            fluxright(j,iNH2,C,Cp,params);
    else
        eq(iNH2) = C(j,iNH2);
    end
    % Equation 11: Oxygen mole fraction
    if j < bound(3)
        eq(iyO2) = stefan_maxwell(2,j,iyO2,C,params);
    elseif j >= bound(3) && j < bound(4)
        dx = mesh(j,C,iregion2,params);
        psiO2 = calc_psiO2(2,j,C,params);
        eq(iyO2) = C(j,iNO2) + (psiO2*(C(j+1,iyO2)-C(j,iyO2)))/dx;
    elseif j >= bound(4)
        eq(iyO2) = C(j,iyO2)+C(j,iyN2)+C(j,iyw)+C(j,iyH2)-1;
    end
    % Equation 12: Nitrogen mole fraction
    if j < bound(4)
        eq(iyN2) = C(j,iyN2);
    elseif j >= bound(4) && j < nj
        eq(iyN2) = stefan_maxwell(2,j,iyN2,C,params);
    elseif j == nj
        cyN2avg = (cyN2in+cyN2out)/2;
        eq(iyN2) = C(j,iyN2)-cyN2avg;
    end
    % Equation 13: Water mole fraction
    if j == 1
        eq(iyw) = C(j,iNw) - fluxright(j,iNw,C,Cp,params);
    elseif j > 1 && j <= bound(3)
        eq(iyw) =  stefan_maxwell(1,j,iyw,C,params);
    elseif j > bound(3) && j < bound(4)
```

```
        eq(iyw) = C(j,iyw);
    elseif j >= bound(4)  && j < nj
        eq(iyw) = stefan_maxwell(2,j,iyw,C,params);
    elseif j == nj
        eq(iyw) = C(j,iNw) - fluxleft(j,iNw,C,Cp,params);
    end
    % Equation 14: Hydrogen mole fraction
    if j <= bound(3)
        eq(iyH2) = (C(j,iyw)+C(j,iyH2)+C(j,iyN2)+C(j,iyO2))-1;
    elseif j > bound(3) && j <= bound(4)
        dx = mesh(j,C,iregion1,params);
        psiH2 = calc_psiH2(1,j,C,params);
        eq(iyH2) = C(j,iNH2) + (psiH2*(C(j,iyH2)-C(j-1,iyH2)))/dx;
    elseif j > bound(4)
        eq(iyH2) = stefan_maxwell(1,j,iyH2,C,params);
    end
    % Equation 16: Pressure
    if j == 1
        eq(ipg) = C(j,ipg) - P0;
    elseif j > 1 && j <= bound(3)
        eq(ipg) = calc_pressure(1,iregion1,j,C,params);
    elseif j > bound (3)  && j < bound(4)
        eq(ipg) = C(j,ipg);
    elseif j >= bound(4)  && j < nj
        eq(ipg) = calc_pressure(2,iregion2,j,C,params);
    elseif j == nj
        eq(ipg) = C(j,ipg) - P0;
    end

else
%% PINHOLE
    % Equation 9: Water flux
    if j == 1
        eq(iNw) = aNwin - aNwDM - aNwout; % gas channel mass balance
    else
        eq(iNw) = fluxleft(j,iNw,C,Cp,params)-...
            fluxright(j,iNw,C,Cp,params);
    end
    % Equation 10: Hydrogen flux
    if j == 1
        eq(iNH2) = C(j,iyw)+C(j,iyH2)+C(j,iyO2)+C(j,iyN2)-1;
    else
        eq(iNH2) = fluxleft(j,iNH2,C,Cp,params)-...
            fluxright(j,iNH2,C,Cp,params);
    end
    % Equation 11: Oxygen mole fraction
    if j < nj
        eq(iyO2) = stefan_maxwell(2,j,iyO2,C,params);
    else
        eq(iyO2) = C(j,iyO2)+C(j,iyN2)+C(j,iyw)+C(j,iyH2)-1;
    end
    % Equation 12: Nitrogen mole fraction
    if j == 1
        eq(iyN2) = stefan_maxwell(2,j,iyN2,C,params);
    elseif j > 1 && j < nj
        eq(iyN2) = C(j,iyO2)+C(j,iyN2)+C(j,iyw)+C(j,iyH2)-1;
    elseif j == nj
```

```
        cyN2avg = (cyN2in+cyN2out)/2;
        eq(iyN2) = C(j,iyN2)-cyN2avg;
    end
    % Equation 13: Water mole fraction
    if j < nj
        eq(iyw) = stefan_maxwell(2,j,iyw,C,params);
    elseif j == nj
        eq(iyw) = cNwin + cNwDM - cNwout;
    end
    % Equation 14: Hydrogen mole fraction
    if j < nj
        eq(iyH2) = stefan_maxwell(2,j,iyH2,C,params);
    else
        eq(iyH2) = C(j,iNH2);
    end
    % Equation 16: Pressure
    if j == 1
        eq(ipg) = C(j,ipg) - P0;
    else
        eq(ipg) = calc_pressure(1,iregion1,j,C,params);
    end
end

% Equation 15: Temperature (energy balance)
    if j == 1 % anode gas channel
        T0 = params(25); % initial temperature (K)
        Tcool = T0;
        [Hgasin,Hgasout] = calc_enthalpy(j,T0,C,[0;0;aywin;ayH2in]);
        aconvin = Hgasin*agasNin;
        agas1D = 0;
        for i = iNO2:iNH2
            agas1D = agas1D + fluxright(j,i,C,Cp,params);
        end
        aconvout = Hgasout*(agasNout+agas1D);
        agenohm = C(j,ii1)*C(j,ii1)*0.05/100;
        sigma = calc_sigma(iregion2,params);
        htcoeff = params(35); % heat transfer coeff (W/cm2 K)
        eq(iT) = agenohm-aconvout+aconvin-htcoeff*(C(j,iT)-Tcool) - ...
           energyfluxright(j,C,Cp,sigma,[],params);
    elseif j > 1 && j < nj
        % anode diffusion media, anode catalyst layer, membrane,
        % cathode catalyst layer, cathode diffusion media
        sigmaL = calc_sigma(iregion1,params);
        sigmaR = calc_sigma(iregion2,params);
        if j >= bound(2) && j <= bound(5)
            [kappaL,~,~,~,~,~] = calc_mem_props(1,j,C,params);
            [kappaR,~,~,~,~,~] = calc_mem_props(2,j,C,params);
        else
            kappaL = [];
            kappaR = [];
        end
        eq(iT) = energyfluxleft(j,C,Cp,sigmaL,kappaL,params) -...
            energyfluxright(j,C,Cp,sigmaR,kappaR,params);
    elseif j == nj
        % cathode gas channel
        T0 = params(25); % initial temperature (K)
        Tin = T0;
        Tcool = T0;
        [Hgasin,Hgasout] = ...
            calc_enthalpy(j,Tin,C,[cyO2in;cyN2in;cywin;0]);
        cconvin = Hgasin*cgasNin;
        cgas1D = 0;
        for i = iNO2:iNH2
            cgas1D = cgas1D + fluxleft(j,i,C,Cp,params);
        end
        cconvout = Hgasout*(cgasNout-cgas1D);
        cgenohm = C(j,ii1)*C(j,ii1)*0.05/100;
        sigma = calc_sigma(iregion1,params);
        htcoeff = params(35); % heat transfer coefficient (W/cm2 K)
        eq(iT) = cgenohm-cconvout+cconvin-htcoeff*(C(j,iT)-Tcool)+...
            energyfluxleft(j,C,Cp,sigma,[],params);
    end

% Equation 17: Membrane Thickness
    if j > bound(3) && j <= bound(4)
        eq(iL) = C(j,iL)-C(j-1,iL);
    elseif j == bound(3)
        eq(iL) = C(j,iL) - C(j,itau);
    else
        eq(iL) = C(j,iL);
    end

% Equation 18: Membrane Expansion Fraction
    if j >= bound(3) && j < bound(4)
        dx = L(3)/(bound(iregion2+1)-bound(iregion2));
        Lmem = calc_Lmem(2,j,C,params);
        eq(itau) = C(j+1,itau)-C(j,itau)+Lmem*dx;
    else
        eq(itau) = C(j,itau);
    end

%% initialize transient
if isempty(Cp) == 1
    if fCe0 == 0
        eq(ifCe) = C(j,ifCe);
        eq(imuCe) = C(j,imuCe);
        eq(iNCe) = C(j,iNCe);
        eq(inCe) = C(j,inCe);
    else
    % Equation 19: Cerium Exchange Fraction
        if j > bound(2) && j <= bound(5)
            eq(ifCe) = fluxleft(j,iNCe,C,Cp,params)-...
                 fluxright(j,iNCe,C,Cp,params);
        else
            eq(ifCe) = C(j,ifCe);
        end
    % Equation 20: Cerium Electrochemical Potential
        if j > bound(2) && j < bound(5)
            CH = (1/zH)*(1-C(j,ifCe))*(rho_m/EW);
            CH1 = (1/zH)*(1-C(j+1,ifCe))*(rho_m/EW);
            CCe = (1/zCe)*C(j,ifCe)*(rho_m/EW);
            CCe1 = (1/zCe)*C(j+1,ifCe)*(rho_m/EW);
            cCe = 0.5*(CCe+CCe1);
            cH = 0.5*(CH+CH1);
            T = 0.5*(C(j,iT)+C(j+1,iT));
            if cCe == 0
                eq(imuCe) = C(j,imuCe) - 1e-4;
            else
                eq(imuCe) = C(j+1,imuCe)-C(j,imuCe) -...
                    ((R*T)/cCe)*(CCe1-CCe)+(zCe/zH)*((R*T)/cH)*(CH1-CH);
            end
        elseif j == bound(5)
            eq(imuCe) = C(j,inCe) - (1/zCe)*fCe0*(rho_m/EW)*L(3);
        else
            eq(imuCe) = C(j,imuCe)-1e-4;
        end
    % Equation 21: Cerium Flux
        if j >= bound(2) && j < bound(5)
            dx = mesh(j,C,iregion2,params);
            [kappa,xi,alphaCe0,tCe,~,alphaCeCe] = ...
                calc_mem_props(2,j,C,params);
            eq(iNCe) = C(j,iNCe) + (alphaCeCe+((tCe/zCe)^2)*...
                (kappa/F^2))*(C(j+1,imuCe)-C(j,imuCe))/dx + ...
                (alphaCe0+(xi*tCe*kappa)/(zCe*F^2))*...
                (C(j+1,imuw)-C(j,imuw))/dx + ((tCe*kappa)/(zCe*F))*...
                (C(j+1,iv2)-C(j,iv2))/dx;
        else
            eq(iNCe) = C(j,iNCe);
        end
    % Equation 22: Total Mass of Cerium
        if j == bound(2)
            eq(inCe) = C(j,inCe);
        elseif j > bound(2) && j <= bound(5)
            dx = mesh(j,C,iregion1,params);
            eq(inCe) = C(j,inCe)-C(j-1,inCe) - ...
                (1/zCe)*(C(j-1,ifCe)*epsM(iregion1)*(rho_m/EW)+...
                C(j,ifCe)*epsM(iregion1)*(rho_m/EW))/2*dx;
        else
            eq(inCe) = C(j,inCe);
        end
    end
if degkin == 1
    eq(icRfSO3) = C(j,icRfSO3);
elseif degkin == 2
    % Equation 25: Membrane Sulfonic Acid Group Concentration
    if j <= bound(2)
        eq(icRfSO3) = C(j,icRfSO3);
    elseif j > bound(2) && j <= bound(5)
        eq(icRfSO3) = C(j,icRfSO3) - (rho_m/EW);
    else
        eq(icRfSO3) = C(j,icRfSO3);
    end
end
% Equation 28: End-Chain Sites
    eq(icCO2H) = C(j,icCO2H);

% Equation 29: Degraded SO3 Side-Chain
    eq(icRfalphaO) = C(j,icRfalphaO);

% Equation 30: Degraded Side-Chain
    eq(icRfbetaO) = C(j,icRfbetaO);
```

```
else
%% Transient
    if fCe0 == 0
        eq(ifCe) = C(j,ifCe);
        eq(imuCe) = C(j,imuCe);
        eq(iNCe) = C(j,iNCe);
        eq(inCe) = C(j,inCe);
    else
        % Equation 19: Cerium Exchange Fraction
        if j > bound(2) && j <= bound(5)
            eq(ifCe) = fluxleft(j,iNCe,C,Cp,params)-...
                 fluxright(j,iNCe,C,Cp,params);
        else
            eq(ifCe) = C(j,ifCe);
        end
        % Equation 20: Cerium Electrochemical Potential
        if j > bound(2) && j < bound(5)
            CH = (1/zH)*(1-C(j,ifCe))*(rho_m/EW);
            CH1 = (1/zH)*(1-C(j+1,ifCe))*(rho_m/EW);
            CCe = (1/zCe)*C(j,ifCe)*(rho_m/EW);
            CCe1 = (1/zCe)*C(j+1,ifCe)*(rho_m/EW);
            cCe = 0.5*(CCe+CCe1);
            cH = 0.5*(CH+CH1);
            T = 0.5*(C(j,iT)+C(j+1,iT));
            if cCe == 0
                eq(imuCe) = C(j,imuCe) - 1e-4;
            else
                eq(imuCe) = C(j+1,imuCe)-C(j,imuCe) -...
                    ((R*T)/cCe)*(CCe1-CCe)+(zCe/zH)*((R*T)/cH)*(CH1-CH);
            end
        elseif j == bound(5)
            fCe0 = params(41);
            eq(imuCe) = C(j,inCe) - (1/zCe)*fCe0*(rho_m/EW)*L(3);
        else
            eq(imuCe) = C(j,imuCe)-1e-4;
        end
        % Equation 21: Cerium Flux
        if j >= bound(2) && j < bound(5)
            dx = mesh(j,C,iregion2,params);
            [kappa,xi,alphaCe0,tCe,~,alphaCeCe] = ...
                calc_mem_props(2,j,C,params);
            eq(iNCe) = C(j,iNCe) + (alphaCeCe+((tCe/zCe)^2)*...
                (kappa/F^2))*(C(j+1,imuCe)-C(j,imuCe))/dx + ...
                (alphaCe0+(xi*tCe*kappa)/(zCe*F^2))*...
                (C(j+1,imuw)-C(j,imuw))/dx + ((tCe*kappa)/(zCe*F))*...
                (C(j+1,iv2)-C(j,iv2))/dx;
        else
            eq(iNCe) = C(j,iNCe);
        end
        % Equation 22: Total Mass of Cerium
        if j == bound(2)
            eq(inCe) = C(j,inCe);
        elseif j > bound(2) && j <= bound(5)
            dx = mesh(j,C,iregion1,params);
            eq(inCe) = C(j,inCe)-C(j-1,inCe) - ...
                (1/zCe)*(C(j-1,ifCe)*epsM(iregion1)*(rho_m/EW)+...
                C(j,ifCe)*epsM(iregion1)*(rho_m/EW))/2*dx;
            if C(j,inCe) < 0
                eq(inCe) = C(j,inCe);
            end
        else
            eq(inCe) = C(j,inCe);
        end
    end

    if degkin == 1
        eq(icRfSO3) = C(j,icRfSO3);
        eq(icCO2H) = C(j,icCO2H);
        eq(icRfalphaO) = C(j,icRfalphaO);
        eq(icRfbetaO) = C(j,icRfbetaO);
    elseif degkin == 2
        % Equation 25: Membrane Sulfonic Acid Group Concentration
        if j <= bound(2)
            eq(icRfSO3) = C(j,icRfSO3);
        elseif j > bound(2) && j <= bound(5)
            eq(icRfSO3) = fluxleft(j,icRfSO3,C,Cp,params)-...
                fluxright(j,icRfSO3,C,Cp,params);
        else
            eq(icRfSO3) = C(j,icRfSO3);
        end
        % Equation 28: End-Chain Sites
        if j <= bound(2)
            eq(icCO2H) = C(j,icCO2H);
        elseif j > bound(2) && j <= bound(5)
            eq(icCO2H) = C(j,icCO2H) - Cp(j,icCO2H);
        else
            eq(icCO2H) = C(j,icCO2H);
        end
        % Equation 29: Degraded SO3 Side-Chain
        if j <= bound(2)
            eq(icRfalphaO) = C(j,icRfalphaO);
        elseif j > bound(2) && j <= bound(5)
            eq(icRfalphaO) = fluxleft(j,icRfalphaO,C,Cp,params)-...
                fluxright(j,icRfalphaO,C,Cp,params);
        else
            eq(icRfalphaO) = C(j,icRfalphaO);
        end

        % Equation 30: Degraded Side-Chain
        if j <= bound(2)
            eq(icRfbetaO) = C(j,icRfbetaO);
        elseif j > bound(2) && j <= bound(5)
            eq(icRfbetaO) = fluxleft(j,icRfbetaO,C,Cp,params)-...
                fluxright(j,icRfbetaO,C,Cp,params);
        else
            eq(icRfbetaO) = C(j,icRfbetaO);
        end
    end
end

if degkin == 1
    eq(iNH2O2) = C(j,iNH2O2);
    eq(icH2O2) = C(j,icH2O2);
    eq(icOH) = C(j,icOH);
```

```
% Equation 26: HF Flux
    if j < 0.5*(bound(4)+bound(5))
        eq(iNHF) = fluxleft(j,iNHF,C,Cp,params)-...
            fluxright(j,iNHF,C,Cp,params);
    elseif j == 0.5*(bound(4)+bound(5))
        eq(iNHF) = C(j,iNHF);
    else
        eq(iNHF) = fluxleft(j,iNHF,C,Cp,params)-...
            fluxright(j,iNHF,C,Cp,params);
    end
% Equation 27: Fluoride Ion Concentration
    DHF_GDL = 0.26;% cm2/s (Wong Kjeang 2014)
    DHF = 1.5e-6; % cm^2s^-1 (Wong & Kjeang 2014)
    if j < bound(2)
        dx = mesh(j,C,iregion2,params);
        eq(icHF) = (C(j+1,icHF)-C(j,icHF))/dx +...
            C(j,iNHF)/DHF_GDL;
    elseif j >= bound(2)  && j < 0.5*(bound(4)+bound(5))
        dx = mesh(j,C,iregion2,params);
        eq(icHF) =  (C(j+1,icHF)-C(j,icHF))/dx +...
            (C(j,icHF)+C(j+1,icHF))*(C(j+1,iNHF)+C(j,iNHF))/(2*DHF);
    elseif j == 0.5*(bound(4)+bound(5))
        eq(icHF) = (C(j+1,icHF)-C(j,icHF))/dx +...
            (C(j,icHF)+C(j+1,icHF))*(C(j+1,iNHF)+C(j,iNHF))/(2*DHF)+...
            (C(j,icHF)-C(j-1,icHF))/dx +...
            (C(j,icHF)+C(j-1,icHF))*(C(j-1,iNHF)+C(j,iNHF))/(2*DHF);
    elseif j > 0.5*(bound(4)+bound(5))  && j < bound(5)
        dx = mesh(j,C,iregion1,params);
        eq(icHF) = (C(j,icHF)-C(j-1,icHF))/dx +...
            (C(j,icHF)+C(j-1,icHF))*(C(j-1,iNHF)+C(j,iNHF))/(2*DHF);
    else
        dx = mesh(j,C,iregion1,params);
        eq(icHF) = (C(j,icHF)-C(j-1,icHF))/dx +...
            (C(j,icHF)+C(j-1,icHF))*(C(j-1,iNHF)+C(j,iNHF))/(2*DHF_GDL);
    end
elseif degkin == 2
% Equation 23: Hydrogen Peroxide Flux
    if j == 1
        eq(iNH2O2) = C(j,icH2O2);
    else
        eq(iNH2O2) = fluxleft(j,iNH2O2,C,Cp,params)-...
            fluxright(j,iNH2O2,C,Cp,params);
    end
% Equation 24: Hydrogen Peroxide Concentration
    if j < bound(2)
        dx = mesh(j,C,iregion2,params);
        DH2O2_GDL = 0.188; % cm2/s (Wong Kjeang 2014)
        eq(icH2O2) = C(j,iNH2O2) + DH2O2_GDL*eps0(iregion2)^1.5*...
            (C(j+1,icH2O2)-C(j,icH2O2))/dx;
    elseif j >= bound(2)  && j < bound(5)
        dx = mesh(j,C,iregion2,params);
        DH2O2 = 1.5e-6; % cm2/s (Gubler)
        eq(icH2O2) = C(j,iNH2O2) + DH2O2*epsM(iregion2)^1.5*...
            (C(j+1,icH2O2)-C(j,icH2O2))/dx;
    elseif j < nj
        dx = mesh(j,C,iregion2,params);
        DH2O2_GDL = 0.188; % cm2/s (Wong Kjeang 2014)
        eq(icH2O2) = C(j,iNH2O2) + DH2O2_GDL*eps0(iregion2)^1.5*...
            (C(j+1,icH2O2)-C(j,icH2O2))/dx;
    else
        eq(icH2O2) = C(j,icH2O2);
    end
% Equation 26: HF Flux
    if j == 1
        eq(iNHF) =  C(j,icHF);
    else
        eq(iNHF) = fluxleft(j,iNHF,C,Cp,params)-...
            fluxright(j,iNHF,C,Cp,params);
    end
% Equation 27: HF Concentration
    if j < bound(2)
        DHF_GDL = 0.26;% cm2/s (Wong Kjeang 2014)
        dx = mesh(j,C,iregion2,params);
        eq(icHF) = C(j,iNHF) + DHF_GDL*eps0(iregion2)^1.5*...
            (C(j+1,icHF)-C(j,icHF))/dx;
    elseif j >= bound(2) && j < bound(5)
        DHF = 1.5e-6; % cm^2s^-1 (Wong & Kjeang 2014)
        dx = mesh(j,C,iregion2,params);
        eq(icHF) = C(j,iNHF) + DHF*epsM(iregion2)^1.5*...
            (C(j+1,icHF)-C(j,icHF))/dx;
    elseif j < nj
        DHF_GDL = 0.26;% cm2/s (Wong Kjeang 2014)
        dx = mesh(j,C,iregion2,params);
        eq(icHF) = C(j,iNHF) + DHF_GDL*eps0(iregion2)^1.5*...
            (C(j+1,icHF)-C(j,icHF))/dx;
    else
        eq(icHF) = C(j,icHF);
    end
% Equation 31: OH radicals concentration
    if j <= bound(2)
        eq(icOH) = C(j,icOH);
    elseif j > bound(2) && j <= bound(5)
        eq(icOH) = fluxleft(j,icOH,C,Cp,params)-...
            fluxright(j,icOH,C,Cp,params);
    else
        eq(icOH) = C(j,icOH);
    end
end
```

```
function NL = fluxleft(j,i,C,Cp,params)
% Calculates the flux exiting the box to the left of point j

ii2 = 3; iNwmem = 6; iNO2 = 7; iNw = 9; iNH2 = 10; iT = 15; ipg = 16;
ifCe = 19; iNCe = 21; iNH2O2 = 23; icH2O2 = 24; icRfSO3 = 25; iNHF = 26;
icHF = 27; icCO2H = 28; icRfalphaO = 29; icRfbetaO = 30; icOH = 31;

% parameters
nspecies = params(3);
bound = params(9:14);
R = 83.14;     % ideal gas constant (cm3 bar/mol K)
F = 96485;     % Faraday's constant (C/mol)
zCe = 3;
EW = params(30); % membrane equivalent weight (g/mol)
rho_m = params(31); % dry membrane density (g/cm3)
fCe0 = params(41); % fraction of SO3- site occupied by Ce
degkin = params(22); % empirical or microkinetics

iregion = region(1,j,bound);
dx = mesh(j,C,iregion,params);

eps_hole = params(40); % pinhole volume fraction
eps0 = params(42:46); % void fractions for gas transport
epsM = params(47:51); % volume fraction of ionomer

% Flux in the box to the left
if i == iNH2
    if eps_hole == 0
        flux = C(j,iNH2);
    else
        flux = C(j-1,iNH2);
    end
elseif i == iNw
    if eps_hole == 0
        if j <= bound(3)
            flux = C(j,i);
        elseif j >= bound(4)
            flux = C(j-1,i);
        end
    else
        flux = C(j-1,i);
    end
elseif i == icRfSO3 || i == icCO2H || i == icRfalphaO ||...
        i == icRfbetaO || i == icOH
    flux = 0;
elseif i == iNHF
    if degkin == 1
        if j <= 0.5*(bound(4)+bound(5))
            flux = C(j,i);
        elseif j > 0.5*(bound(4)+bound(5))
            flux = C(j-1,i);
        end
    elseif degkin == 2
        flux = C(j-1,i);
    end
else
    flux = C(j-1,i);
```

```
end

w = 0.75;
% Reaction terms
% reaction 1 = HOR
% reaction 2 = 4e- ORR
% reaction 3 = water transfer from membrane
% reaction 4 = 2e- ORR
% reaction 5 = peroxide radical formation
% reaction 6 = hydroxyl radical attack on membrane side chain
% reaction 7 = hydroxyl radical attack on membrane end chain
% reaction 8 = degradation SO3 product reaction
% reaction 9 = degradation product reaction
% reaction 10 = Cerium Quenching of hydroxyl
st = zeros(31,10); % stoichiometric coefficients
st(iNO2,:) = [0 -1 0 -1 0 0 0 0 0 0];
st(iNwmem,:) = [0 2 1 0 0 0 0 0 -2 1];
st(iNH2,:) = [-1 0 0 0 0 0 0 0 0 0];
st(iNw,:) = [0 0 -1 0 0 0 0 0 0 0];
st(ii2,:) = [2*F -4*F 0 -2*F 0 0 0 0 0 -F];
st(iNH2O2,:) = [0 0 0 1 -1 0 0 0 0 0];
st(icRfSO3,:) = [0 0 0 0 0 -1 0 0 0 0];
if degkin == 1
    st(iNHF,:) = [0 0 0 0.4/dx 0 0 0 0 0 0];
elseif degkin == 2
    st(iNHF,:) = [0 0 0 0 0 4 4 6 3 0];
end
st(icCO2H,:) = [0 0 0 0 0 0 0 0 2 0];
st(icRfalphaO,:) = [0 0 0 0 0 1 0 -1 0 0];
st(icRfbetaO,:) = [0 0 0 0 0 0 1 0 -1 0];
st(icOH,:) = [0 0 0 0 2 0 0 0 0 -1];
rate = react(j,C,iregion,params);
if j > 1
    rateL = react(j-1,C,iregion,params);
else
    rateL = rate;
end
gen = st(i,:)*(w*rate+(1-w)*rateL)*dx/2;

acc = 0;
if isempty(Cp) == 0  %transient
    dt = params(77); % time spacing
    if i == ii2
        acc = 0;
    elseif i == iNwmem
        [~,~,C0V] = calc_lambda(j,C,params);
        [~,~,C0VL] = calc_lambda(j-1,C,params);
        [~,~,C0Vp] = calc_lambda(j,Cp,params);
        [~,~,C0VpL] = calc_lambda(j-1,Cp,params);
        dcdt = w*0.5*(C0V-C0Vp)/dt+(1-w)*0.5*(C0VL-C0VpL)/dt;
        acc = epsM(iregion)*dcdt*dx/2;
    elseif i == iNCe
        if fCe0 > 0
            CCe = (1/zCe)*C(j,ifCe)*(rho_m/EW);
            CCe1 = (1/zCe)*C(j-1,ifCe)*(rho_m/EW);
            CCep = (1/zCe)*Cp(j,ifCe)*(rho_m/EW);
            CCe1p = (1/zCe)*Cp(j-1,ifCe)*(rho_m/EW);
            dcdt = w*0.5*(CCe-CCep)/dt+(1-w)*0.5*(CCe1-CCe1p)/dt;
            acc = epsM(iregion)*dcdt*dx/2;
        end
    elseif i == iNH2O2
        dcdt = w*0.5*(C(j,icH2O2)-Cp(j,icH2O2))/dt+...
            (1-w)*0.5*(C(j-1,icH2O2)-Cp(j-1,icH2O2))/dt;
        acc = dcdt*dx/2;
    elseif i == icRfSO3
        dcdt = w*0.5*(C(j,icRfSO3)-Cp(j,icRfSO3))/dt+...
            (1-w)*0.5*(C(j-1,icRfSO3)-Cp(j-1,icRfSO3))/dt;
        acc = dcdt*dx/2;
    elseif i == icCO2H
        dcdt = w*0.5*(C(j,icCO2H)-Cp(j,icCO2H))/dt+...
            (1-w)*0.5*(C(j-1,icCO2H)-Cp(j-1,icCO2H))/dt;
        acc = dcdt*dx/2;
    elseif i == icRfalphaO
        dcdt = w*0.5*(C(j,icRfalphaO)-Cp(j,icRfalphaO))/dt+...
            (1-w)*0.5*(C(j-1,icRfalphaO)-Cp(j-1,icRfalphaO))/dt;
        acc = dcdt*dx/2;
    elseif i == icRfbetaO
        dcdt = w*0.5*(C(j,icRfbetaO)-Cp(j,icRfbetaO))/dt+...
            (1-w)*0.5*(C(j-1,icRfbetaO)-Cp(j-1,icRfbetaO))/dt;
        acc = dcdt*dx/2;
    elseif i == icOH
        dcdt = w*0.5*(C(j,icOH)-Cp(j,icOH))/dt+...
            (1-w)*0.5*(C(j-1,icOH)-Cp(j-1,icOH))/dt;
        acc = dcdt*dx/2;
    elseif i == iNHF
        if j == 1
            dcdt = (C(j,icHF)-Cp(j,icHF))/dt;
        else
            dcdt = w*0.5*(C(j,icHF)-Cp(j,icHF))/dt+...
                (1-w)*0.5*(C(j-1,icHF)-Cp(j-1,icHF))/dt;
        end
        acc = dcdt*dx/2;
    else
        if j == 1
            CT = C(j,ipg)/(C(j,iT)*R);
            CTp = Cp(j,ipg)/(Cp(j,iT)*R);
            dcdt = (CT*C(j,i+nspecies)-CTp*Cp(j,i+nspecies))/dt;
        else
            CT = (C(j,ipg)/C(j,iT)+C(j-1,ipg)/C(j-1,iT))/(2*R);
            CTp = (Cp(j,ipg)/Cp(j,iT)+Cp(j-1,ipg)/Cp(j-1,iT))/(2*R);
            dcdt = w*0.5*(CT*C(j,i+nspecies)-...
                CTp*Cp(j,i+nspecies))/dt+...
                (1-w)*0.5*(CT*C(j-1,i+nspecies)-...
                CTp*Cp(j-1,i+nspecies))/dt;
        end
        acc = eps0(iregion)*dcdt*dx/2;
    end
end

NL = flux + gen - acc;
end
```

```matlab
function NR = fluxright(j,i,C,Cp,params)
% Calculates the flux exiting the box to the right of point j

ii2 = 3; iNwmem = 6; iNO2 = 7; iNw = 9; iNH2 = 10; iT = 15; ipg = 16;
ifCe = 19; iNCe = 21; iNH2O2 = 23; icH2O2 = 24; icRfSO3 = 25; iNHF = 26;
icHF = 27; icCO2H = 28; icRfalphaO = 29; icRfbetaO = 30; icOH = 31;

% parameters
nspecies = params(3);
bound = params(9:14); nj = bound(6);
R = 83.14;     % ideal gas constant (cm3 bar/mol K)
F = 96485;     % Faraday's constant (C/mol)
zCe = 3;
EW = params(30); % membrane equivalent weight (g/mol)
rho_m = params(31); % dry membrane density (g/cm3)
fCe0 = params(41); % fraction of SO3- site occupied by Ce
degkin = params(22); % empirical or microkinetics
iregion = region(2,j,bound);
dx = mesh(j,C,iregion,params);
eps_hole = params(40); % pinhole volume fraction
eps0 = params(42:46); % void fractions for gas transport
epsM = params(47:51); % volume fraction of ionomer

% Flux in the box to the right
if i == iNH2
    if eps_hole == 0
        flux = C(j+1,iNH2);
    else
        flux = C(j,iNH2);
    end
elseif i == iNw
    if eps_hole == 0
        if j <= bound(3)
            flux = C(j+1,i);
        elseif j >= bound(4)
            flux = C(j,i);
        end
    else
        flux = C(j,i);
    end
elseif i == icRfSO3 || i == icCO2H || i == icRfalphaO ||...
        i == icRfbetaO || i == icOH
    flux = 0;
elseif i == iNHF
    if degkin == 1
        if j <= 0.5*(bound(4)+bound(5))
            flux = C(j+1,i);
        elseif j > 0.5*(bound(4)+bound(5))
            flux = C(j,i);
        end
    elseif degkin == 2
        flux = C(j,i);
    end
else
    flux = C(j,i);
end
w = 0.75;
```

```
% Reaction terms
% reaction 1 = HOR
% reaction 2 = 4e- ORR
% reaction 3 = water transfer from membrane
% reaction 4 = 2e- ORR
% reaction 5 = peroxide radical formation
% reaction 6 = hydroxyl radical attack on membrane side chain
% reaction 7 = hydroxyl radical attack on membrane end chain
% reaction 8 = degradation SO3 product reaction
% reaction 9 = degradation product reaction
% reaction 10 = Cerium Quenching of hydroxyl
st = zeros(31,10);  % stoichiometric coefficients
st(iNO2,:) = [0 -1 0 -1 0 0 0 0 0 0];
st(iNwmem,:) = [0 2 1 0 0 0 0 0 -2 1];
st(iNH2,:) = [-1 0 0 0 0 0 0 0 0 0];
st(iNw,:) = [0 0 -1 0 0 0 0 0 0 0];
st(ii2,:) = [2*F -4*F 0 -2*F 0 0 0 0 0 -F];
st(iNH2O2,:) = [0 0 0 1 -1 0 0 0 0 0];
st(icRfSO3,:) = [0 0 0 0 0 -1 0 0 0 0];
if degkin == 1
    st(iNHF,:) = [0 0 0 0.4/dx 0 0 0 0 0 0];
elseif degkin == 2
    st(iNHF,:) = [0 0 0 0 0 4 4 6 3 0];
end
st(icCO2H,:) = [0 0 0 0 0 0 0 0 2 0];
st(icRfalphaO,:) = [0 0 0 0 0 1 0 -1 0 0];
st(icRfbetaO,:) = [0 0 0 0 0 0 1 0 -1 0];
st(icOH,:) = [0 0 0 0 2 0 0 0 0 -1];
rate = react(j,C,iregion,params);
if j < nj
    rateR = react(j+1,C,iregion,params);
else
    rateR = rate;
end
gen = st(i,:)*(w*rate+(1-w)*rateR)*dx/2;

acc = 0;
if isempty(Cp) == 0 % transient
    dt = params(77); % time spacing
    if i == ii2
        acc = 0;
    elseif i == iNwmem
        [~,~,C0V] = calc_lambda(j,C,params);
        [~,~,C0VR] = calc_lambda(j+1,C,params);
        [~,~,C0Vp] = calc_lambda(j,Cp,params);
        [~,~,C0VpR] = calc_lambda(j+1,Cp,params);
        dcdt = w*0.5*(C0V-C0Vp)/dt+(1-w)*0.5*(C0VR-C0VpR)/dt;
        acc = epsM(iregion)*dcdt*dx/2;
    elseif i == iNCe
        if fCe0 > 0
            CCe = (1/zCe)*C(j,ifCe)*(rho_m/EW);
            CCe1 = (1/zCe)*C(j+1,ifCe)*(rho_m/EW);
            CCep = (1/zCe)*Cp(j,ifCe)*(rho_m/EW);
            CCe1p = (1/zCe)*Cp(j+1,ifCe)*(rho_m/EW);
            dcdt = w*0.5*(CCe-CCep)/dt+(1-w)*0.5*(CCe1-CCe1p)/dt;
            acc = epsM(iregion)*dcdt*dx/2;
        end
    elseif i == iNH2O2
        if j == nj
             dcdt = (C(j,icH2O2)-Cp(j,icH2O2))/dt;
        else
            dcdt = w*0.5*(C(j,icH2O2)-Cp(j,icH2O2))/dt+...
                (1-w)*0.5*(C(j+1,icH2O2)-Cp(j+1,icH2O2))/dt;
        end
        acc = dcdt*dx/2;
    elseif i == icRfSO3
        dcdt = w*0.5*(C(j,icRfSO3)-Cp(j,icRfSO3))/dt+...
            (1-w)*0.5*(C(j+1,icRfSO3)-Cp(j+1,icRfSO3))/dt;
        acc = dcdt*dx/2;
    elseif i == icCO2H
        dcdt = w*0.5*(C(j,icCO2H)-Cp(j,icCO2H))/dt+...
            (1-w)*0.5*(C(j+1,icCO2H)-Cp(j+1,icCO2H))/dt;
        acc = dcdt*dx/2;
    elseif i == icRfalphaO
        dcdt = w*0.5*(C(j,icRfalphaO)-Cp(j,icRfalphaO))/dt+...
            (1-w)*0.5*(C(j+1,icRfalphaO)-Cp(j+1,icRfalphaO))/dt;
        acc = dcdt*dx/2;
    elseif i == icRfbetaO
        dcdt = w*0.5*(C(j,icRfbetaO)-Cp(j,icRfbetaO))/dt+...
            (1-w)*0.5*(C(j+1,icRfbetaO)-Cp(j+1,icRfbetaO))/dt;
        acc = dcdt*dx/2;
    elseif i == icOH
        dcdt = w*0.5*(C(j,icOH)-Cp(j,icOH))/dt+...
            (1-w)*0.5*(C(j+1,icOH)-Cp(j+1,icOH))/dt;
        acc = dcdt*dx/2;
    elseif i == iNHF
        if j == nj
            dcdt = (C(j,icHF)-Cp(j,icHF))/dt;
        else
            dcdt = w*0.5*(C(j,icHF)-Cp(j,icHF))/dt+...
                (1-w)*0.5*(C(j+1,icHF)-Cp(j+1,icHF))/dt;
        end
        acc = dcdt*dx/2;
    else
        if j == nj
            CT = C(j,ipg)/(C(j,iT)*R);
            CTp = Cp(j,ipg)/(Cp(j,iT)*R);
            dcdt = (CT*C(j,i+nspecies)-CTp*Cp(j,i+nspecies))/dt;
        else
            CT = (C(j,ipg)/C(j,iT)+C(j+1,ipg)/C(j+1,iT))/(2*R);
            CTp = (Cp(j,ipg)/Cp(j,iT)+Cp(j+1,ipg)/Cp(j+1,iT))/(2*R);
            dcdt = w*0.5*(CT*C(j,i+nspecies)-
                CTp*Cp(j,i+nspecies))/dt+...
                (1-w)*0.5*(CT*C(j+1,i+nspecies)-...
                CTp*Cp(j+1,i+nspecies))/dt;
        end
        acc = eps0(iregion)*dcdt*dx/2;
    end
end
NR = flux - gen + acc;
end
```

```
function heat = heat_react(j,iregion,C,sigma,kappa,params)
% Function for handling generation of heat from reactions and
% ohmic heating
ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4;
iyO2 = 11; iyH2 = 14; iT = 15; ipg = 16; ifCe = 19;

heat = [0;0;0];
if iregion == 1 || iregion == 5
    % Ohmic Heating
    heat(1) =  C(j,ii1)*C(j,ii1)/sigma;
elseif iregion == 3
    % Ohmic Heating
    heat(1) = C(j,ii2)*C(j,ii2)/kappa;
elseif iregion == 2 || iregion == 4
    F = 96485;                  % Faraday's constant (C/mol)
    R = 8.314;                  % ideal gas constant (J/mol K)
    Tref = 303.15;              % reference temperature (K)
    FRT = F/(R*C(j,iT));
    a12 = params(34);      % electrode specific interfacial area (1/cm)
    phimtH2 = params(35); % thiele mass transfer for H2 (bar cm3 s/mol)
    phimtO2 = params(36); % thiele mass transfer for O2 (bar cm3 s/mol)

    % Ohmic Heating
    heat(1) = C(j,ii1)*C(j,ii1)/sigma + C(j,ii2)*C(j,ii2)/kappa;

    % Heat of Reaction (HOR)
    EAHOR = 9500; % activation energy (J/mol)
    i0HOR = 1e-3*exp((EAHOR/R)*(1/Tref-1/C(j,iT)));
    etaHOR = C(j,iv1)-C(j,iv2); % overpotential
    alphaa = 1;
    alphac = 1;
    kHOR = (i0HOR/(2*F))*(exp(alphaa*FRT*etaHOR));
    phiHOR = sqrt(phimtH2*kHOR);
    effHOR = (3/(phiHOR^2))*(phiHOR/tanh(phiHOR)-1);
    rateHOR = effHOR*(a12/(2*F))*i0HOR*(C(j,ipg)*C(j,iyH2)*...
        exp(alphaa*FRT*etaHOR) -...
        (1-C(j,ifCe))^2*exp(-alphac*FRT*etaHOR));
    peltHOR = -0.013*C(j,iT)/298.15;
    heat(2) = (etaHOR+peltHOR)*rateHOR;

    % Heat of Reaction (ORR)
    EAORR = 73269; % activation energy (J/mol)
    U0 = 4.1868*(70650+8*C(j,iT)*log(C(j,iT))-92.84*C(j,iT))/(2*F);
    i0ORR = 1.1e-8*exp((EAORR/R)*(1/Tref-1/C(j,iT)));
    etaORR = C(j,iv1) - C(j,iv2) - U0; % overpotential
    kORR = (i0ORR/(4*F))*(exp(-alphac*FRT*etaORR));
    phiORR = sqrt(phimtO2*kORR);
    effORR = (3/(phiORR^2))*(phiORR/tanh(phiORR)-1);
    rateORR = effORR*(a12/(4*F))*i0ORR*(C(j,ipg)*C(j,iyO2)*...
        (1-C(j,ifCe))^4*exp(-alphac*FRT*etaORR));
    peltORR = -0.226*C(j,iT)/298.15;
    heat(3) = (etaORR+peltORR)*rateORR;
end
end
```

```
function rate = react(j,C,iregion,params)
% Function for handling homoegenous reactions
rate = zeros(10,1);

iv1 = 2; iv2 = 4; imuw = 5; iyO2 = 11; iyw = 13; iyH2 = 14; iT = 15;
ipg = 16; ifCe = 19; icH2O2 = 24; icRfSO3 = 25; icCO2H = 28;
icRfalphaO = 29; icRfbetaO = 30; icOH = 31;

F = 96485;              % Faraday's constant (C/mol)
R = 8.314;              % ideal gas constant (J/mol K)
Tref = 303.15;          % reference temperature (K)
FRT = F/(R*C(j,iT));
a12 = params(34);      % electrode specific interfacial area (1/cm)
phimtH2 = params(35); % thiele mass transfer for H2 (bar cm3 s/mol)
phimtO2 = params(36); % thiele mass transfer for O2 (bar cm3 s/mol)
kw = params(37);       % water vapor/membrane mass transfer coefficient
fCe0 = params(41);     % fraction of SO3- site occupied by Ce
degkin = params(22);   % empirical or microkinetics

% Electrode reactions and water transport
if iregion == 2 || iregion == 4
    % Hydrogen Oxidation Reaction
    % H2 -> 2H^+ + 2e^-
    alphac = 1; alphaa = 1;
    EAHOR = 9500; % activation energy (J/mol)
    % exchange current density (A/cm2)
    i0HOR = 1e-3*exp((EAHOR/R)*(1/Tref-1/C(j,iT)));
    etaHOR = C(j,iv1)-C(j,iv2); % overpotential
    kHOR = (i0HOR/(2*F))*(exp(alphaa*FRT*etaHOR));
    phiHOR = sqrt(phimtH2*kHOR);
    effHOR = (3/(phiHOR^2))*(phiHOR/tanh(phiHOR)-1);
    rate(1) = effHOR*(a12/(2*F))*i0HOR*(C(j,ipg)*C(j,iyH2)*...
        exp(alphaa*FRT*etaHOR) -...
        (1-C(j,ifCe))^2*exp(-alphac*FRT*etaHOR));
    % Oxygen Reduction Reaction
    % O2 + 4H^+ + 4e^- -> 2H2O
    EAORR = 73269; % activation energy (J/mol)
    U0 = 4.1868*(70650+8*C(j,iT)*log(C(j,iT))-92.84*C(j,iT))/(2*F);
    i0ORR = 1.1e-8*exp((EAORR/R)*(1/Tref-1/C(j,iT)));
    etaORR = etaHOR - U0; % overpotential
    kORR = (i0ORR/(4*F))*(exp(-alphac*FRT*etaORR));
    phiORR = sqrt(phimtO2*kORR);
    effORR = (3/(phiORR^2))*(phiORR/tanh(phiORR)-1);
    rate(2) = effORR*(a12/(4*F))*i0ORR*...
       (C(j,ipg)*C(j,iyO2)*(1-C(j,ifCe))^4*exp(-alphac*FRT*etaORR));
    % water transfer to/from membrane
    % Nw -> Nwmem
    pvap0 = exp(11.6832 - 3816.44/(C(j,iT)-46.13));
    MW = 18.016;       % molecular weight of water (g/mol)
    rho_w = 1.1603-5.371e-4*C(j,iT); % density of water (g/cm3)
    V0 = MW/rho_w;     % molar volume of water (cm3/mol)
    rate(3) = kw*(C(j,imuw) - 0.1*V0*C(j,ipg) - ...
        R*C(j,iT)*log((C(j,iyw)*C(j,ipg))/pvap0));
    % Hydrogen Peroxide Formation (ORR 2e-)
    % O2 + 2H^+ + 2e^- -> H2O2
    U0_e2 = 0.695; % standard potential (V vs. SHE)
    etaH2O2 = etaHOR-U0_e2; % overpotential
    i0H2O2 = 1e-5; %exchange current density (A/cm2)
    effH2O2 = 1;
    if fCe0 > 0
        rate(4) = (a12/(2*F))*effH2O2*i0H2O2*(C(j,ipg)*C(j,iyO2)*...
            (1-C(j,ifCe))^2*exp(-alphac*FRT*etaH2O2));
    else
        rate(4) = (a12/(2*F))*effH2O2*i0H2O2*(C(j,ipg)*C(j,iyO2)*...
            exp(-alphac*FRT*etaH2O2));
    end
end

% Chemical degradation reactions
if degkin == 2
    if iregion == 2 || iregion == 3 || iregion == 4
        EW = params(30); % membrane equivalent weight (g/mol)
        rho_m = params(31); % dry membrane density (g/cm3)
        zCe = 3;
        cCe = (1/zCe)*C(j,ifCe)*(rho_m/EW);

        % rate constants
        k(1) = 3.7e6; % M^-1 s^-1
        k(2) = 5.8e6; % M^-1 s^-1
        k(4) = 3.75e7; % M^-1 s^-1
        k(5) = 7.5e7; % M^-1 s^-1
        k(6) = 1e11; % M^-1 s^-1
        k = k*1000; % convert to cm3/mol/s
        k(3) = 0.003; % s^-1

        % Hydroxyl radical formation
        % H2O2 -> HOrad
        rate(5) = k(3)*C(j,icH2O2);
        % Hydroxyl radical attack on membrane
        % HOrad + RfSO3 -> RfalphaO + 4HF
        rate(6) = k(1)*C(j,icRfSO3)*C(j,icOH);
        % Hydroxyl radical attack on membrane end-chain
        % HOrad + RfCF2COOH -> RfCF2* + 2HF
        rate(7) = k(2)*C(j,icCO2H)*C(j,icOH);
        % Hydroxyl radical attack on membrane side-chain
        % RfalphaO + 3OHrad -> RfbetaO + 6HF
        rate(8) = k(4)*C(j,icRfalphaO)*C(j,icOH);
        % Hydroxyl radical attack on membrane side-chain
        % RfbetaO + 2H2O + OHrad -> 2RfCOOH + 3HF
        rate(9) = k(5)*C(j,icRfbetaO)*C(j,icOH);
        % Cerium quenching of hydroxyl
        % Ce3+ OHrad + H+ -> Ce4+ + H2O
        rate(10) = k(6)*cCe*C(j,icOH);
    end
end
end
```

```
function stefmax = stefan_maxwell(mode,j,i,C,params)
% Calculates the Stefan-Maxwell equation for species fluxes at a point j
% in the gas phase at steady state for constant T and P.
% gas phase species
% 1 = oxygen, 2 = nitrogen, 3 = water, 4 = hydrogen
iNO2 = 7; iNH2 = 10; iyO2 = 11; iyH2 = 14; iT = 15; ipg = 16;
nspecies = params(3); bound = params(9:14);
R = 83.14;    % ideal gas constant (cm3 bar/mol K)
eps0 = params(42:46); % void fractions for gas transport
rad = params(67:71); % characteristic pore size (um)
tau = eps0.^-0.5; % tortuosity
tau(3) = 1;
MW = [31.9988; 28.014; 18.0152; 2.0159]; % (O2 N2 H2O H2)
diffusion = 0;

if mode == 1
    iregion = region(1,j,bound); dx = mesh(j,C,iregion,params);
    T = (C(j,iT)+C(j-1,iT))/2;  pg = (C(j,ipg)+C(j-1,ipg))/2;
    CT = (C(j,ipg)/C(j,iT)+C(j-1,ipg)/C(j-1,iT))/2/R;
    gasmass = ((C(j,iyO2:iyH2)+C(j-1,iyO2:iyH2))/2)*MW;
    drive = (C(j,i)-C(j-1,i))/dx + ((C(j,i)+C(j-1,i))/2)*...
        ((C(j,ipg)-C(j-1,ipg))/dx)*(1-MW(i-iNH2)/gasmass)/pg;
    D = diffcoeff(pg,T);  Deff = D*(eps0(iregion)/tau(iregion));
    Dk = knudsen(T,MW(i-iNH2),rad(iregion));
    Dkeff = Dk*(eps0(iregion)/tau(iregion));
    for k = iNO2:iNO2+nspecies-1
        if k ~= i-nspecies
            diffusion = diffusion + (C(j,i)*C(j,k)+C(j-1,i)*C(j-1,k)...
                -C(j,k+nspecies)*C(j,i-nspecies)-C(j-1,k+nspecies)*...
                C(j-1,i-nspecies))/(2*CT*Deff(k-iNO2+1,i-iyO2+1));
        end
    end
    dk = -(C(j,i-nspecies))/(CT*Dkeff);
    diffusion = diffusion + dk;
elseif mode == 2
    iregion = region(2,j,bound); dx = mesh(j,C,iregion,params);
    T = (C(j,iT)+C(j+1,iT))/2;   pg = (C(j,ipg)+C(j+1,ipg))/2;
    CT = (C(j,ipg)/C(j,iT)+C(j+1,ipg)/C(j+1,iT))/2/R;
    gasmass = ((C(j,iyO2:iyH2)+C(j+1,iyO2:iyH2))/2)*MW;
    drive = (C(j+1,i)-C(j,i))/dx + ((C(j+1,i)+C(j,i))/2)*...
        ((C(j+1,ipg)-C(j,ipg))/dx)*(1-MW(i-iNH2)/gasmass)/pg;
    D = diffcoeff(pg,T);  Deff = D*(eps0(iregion)/tau(iregion));
    Dk = knudsen(T,MW(i-iNH2),rad(iregion));
    Dkeff = Dk*(eps0(iregion)/tau(iregion));
    for k = iNO2:iNO2+nspecies-1
        if k ~= i-nspecies
            diffusion = diffusion + (C(j,i)*C(j,k)+C(j+1,i)*C(j+1,k)...
                -C(j,k+nspecies)*C(j,i-nspecies)-C(j+1,k+nspecies)*...
                C(j+1,i-nspecies))/(2*CT*Deff(k-iNO2+1,i-iyO2+1));
        end
    end
    dk = -(C(j,i-nspecies))/(CT*Dkeff);
    diffusion = diffusion + dk;
end
stefmax = drive - diffusion;
end
```

```
function visc = viscgas(T,yi)
% Gas viscosity given in bar-s (Stiel and Thodos and Bromley
% and Wilke), see Perry's p. 2-363 (B-W is wrong in Perry's)
%      O2        N2       H2O      H2
Tc = [154.8     126.2    647.4    33.3];
xi = [0.0301    0.0407   0.0192   0.230];
MW = [31.9988   28.014   18.015   2.0159];

% Calculate pure gas viscosity (centipoise)
% valid at moderate presures (0.2 atm - 5 atm)
visg = zeros(1,4);
for i=1:2
    if T/Tc(i) <= 1.5
        visg(i)=34e-5/xi(i)*(T/Tc(i))^0.94;
    else
        visg(i)=17.78e-5/xi(i)*(4.58*T/Tc(i)-1.67)^(5/8);
    end
end
visg(3) = (7.55*T/Tc(3)-0.55)*1e-5/xi(3)/0.231^(5/4);
visg(4) = 90.71e-5*(0.1375*T-1.67)^(5/8);
Qij = zeros(4);
% Calculate interaction parameters (eq. 15 Bromley & Wilke 1951)
for i = 1:4
    for j = 1:4
        Qij(i,j)=(1+(visg(i)/visg(j))^0.5*(MW(j)/MW(i))^0.25)^2/...
            sqrt(8)/(1+MW(i)/MW(j))^0.5;
    end
    Qij(i,i) = 0;
end
% Calculate gas-mixture viscosity (eq. 14 Bromley & Wilke 1951)
visc = 0;
sum1 = zeros(1,4);
for i = 1:4
    if abs(yi(i)) > 1e-20
        for j = 1:4
            sum1(i) = yi(j)*Qij(i,j);
            % 1e-8 converts cp to bar-s
            visc = visc + (1e-8*visg(i))/(1+1/yi(i)*sum1(i));
        end
    end
end

end

```

B.4 Mechanical Degradation Model with Multiphase Phenomena

```
function eq = eqn(j,jp,k,dC,C,Cp,params)

C(jp,k) = C(jp,k)+dC;

ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; imuw = 5; iNwmem = 6; iNO2 = 7;
iNN2 = 8; iNw = 9; iNH2 = 10; iyO2 = 11; iyN2 = 12; iyw = 13;
iyH2 = 14; iT = 15; ipg = 16; iL = 17; itau = 18; ipl = 19;
iplmem = 20; iNwl = 21; iNF = 22; icF = 23;

nregion = params(1);
bound = params(2*nregion+3:3*nregion+3);
L = params(nregion+3:2*nregion+2);
nj = bound(6);

F = 96485; % Faraday's constant (C/mol)
R = 8.314; % ideal gas constant (J/mol K)
R1 = 83.14; % ideal gas constant (cm3 bar/mol K)

% calculate properties
iregion1 = region(1,j,params);
iregion2 = region(2,j,params);
[pore1,props1,pore2L,props2L,pore2R,props2R,pore3,props3] =...
    calc_props(j,C,params);

%% Gas Channel Mass Balances
if j == 1 % anode gas channel
    % feed gas (can include N2 as inert)
    % if yH2 = 1, then pure H2 feed
    yH2 = 1;
    % mole fractions in
    p0 = params(3*nregion+5);        % pressure (bar)
    RHa = params(3*nregion+7);       % relative humidity
    Pwsat = params(3*nregion+9);     % water vapor pressure (bar)
    aywin = RHa*Pwsat/p0;
    ayH2in = (1-aywin)*yH2;
    ayN2in = (1-aywin)*(1-yH2);
    % total dry gas flow in (based on stoichiometry)
    lfeed = params(3*nregion+13);    % feed stoichiometry
    flowmode = params(3*nregion+17);
    if flowmode == 1
        adgNin = (C(j,ii1)/(2*F))*(lfeed/yH2);
    else
        adgNin = lfeed*C(j,ipg)/(R1*C(j,iT))/60;
    end
    % gas flows in
    aNwin = adgNin*RHa*Pwsat/(p0-RHa*Pwsat);
    aNN2in = adgNin*(1-yH2);
    % total gas flow in
    agasNin = aNwin+adgNin;
    % liquid flow in
    aNwlin = 0;
    % total dry gas flow out
    adgNout = adgNin - C(j,iNH2) - C(j,iNN2);
    % mole fractions and gas flows out
    if C(j,iyw) < Pwsat/C(j,ipg)
        aywout = C(j,iyw);
    else
        aywout = Pwsat/C(j,ipg);
    end
    aNwDM = fluxright(j,iNw,C,Cp,params,pore2R,pore3);
    aNwlDM = fluxright(j,iNwl,C,Cp,params,pore2R,pore3);
    aNN2out = aNN2in - C(j,iNN2);
    aNwout = adgNout*aywout/(1-aywout);
    ayN2out = aNN2out/(aNwout+adgNout);
    aNwlout = aNwlin + aNwin - aNwDM - aNwlDM - aNwout;
    % total gas flow out
    agasNout = aNwout+adgNout;
    % condensation
    aNcond = aNwlout-aNwlin+aNwlDM;
elseif j == nj % cathode gas channel
    % feed gas (air)
    yO2 = 0.21;
    yN2 = 0.79;
    % mole fractions in
    p0 = params(3*nregion+5);       % pressure (bar)
    RHc = params(3*nregion+8);      % relative humidity
    Pwsat = params(3*nregion+9);    % water vapor pressure (bar)
    cywin = RHc*Pwsat/p0;
    cyO2in = (1-cywin)*yO2;
    cyN2in = (1-cywin)*yN2;
    % total dry gas flow in (based on stoichiometry)
    lair = params(3*nregion+14);    % air stoichiometry
    flowmode = params(3*nregion+17);
    if flowmode == 1
        cdgNin = (C(j,ii1)/(4*F))*(lair/cyO2in);
    else
        cdgNin = lair*C(j,ipg)/(R1*C(j,iT))/60;
    end
    % gas flows in
    cNwin = cdgNin*RHc*Pwsat/(p0-RHc*Pwsat);
    cNN2in = cdgNin*yN2;
    % total gas flow in
    cgasNin = cNwin+cdgNin;
    % liquid flow in
    cNwlin = 0;
    % total dry gas flow out
    cdgNout = cdgNin+C(j,iNO2)+C(j,iNN2);
    % mole fractions and gas flows out
    if C(j,iyw) < Pwsat/C(j,ipg)
        cywout = C(j,iyw);
    else
        cywout = Pwsat/C(j,ipg);
    end
    cNwDM = fluxleft(j,iNw,C,Cp,params,pore2L,pore1);
    cNwlDM = fluxleft(j,iNwl,C,Cp,params,pore2L,pore1);
    cNwout = cdgNout*cywout/(1-cywout);
    cNN2out = cNN2in+C(j,iNN2);
    cyN2out = cNN2out/(cNwout+cdgNout);
    cNwlout = cNwlin+cNwin+cNwDM+cNwlDM-cNwout;
    % total gas flow out
    cgasNout = cNwout+cdgNout;
    % condensation
    cNcond = cNwlout-cNwlin-cNwlDM;
end

%% Governing Equations

% Equation 1: solid phase current
    if j <= bound(3)
        eq(ii1) = (C(j+1,ii2)-C(j,ii2))+(C(j+1,ii1)-C(j,ii1));
    elseif j > bound(3) && j < bound(4)
        eq(ii1) = C(j,ii1);
    else
        eq(ii1) = (C(j,ii1)-C(j-1,ii1))+(C(j,ii2)-C(j-1,ii2));
    end
% Equation 2: solid phase potential
    if j == 1
        eq(iv1) = C(j,iv1);
    elseif j > 1 && j <= bound(3)
        dx = mesh(j,C,iregion1,params);
        condL = 0.5*(props1(11)+props2L(11));
        eq(iv1) = C(j,ii1) + condL*(C(j,iv1)-C(j-1,iv1))/dx;
    elseif j > bound(3) && j < bound(4)
        eq(iv1) = C(j,iv1);
    elseif j >= bound(4) && j < nj
        dx = mesh(j,C,iregion2,params);
        condR = 0.5*(props3(11)+props2R(11));
        eq(iv1) = C(j,ii1) + condR*(C(j+1,iv1)-C(j,iv1))/dx;
    elseif j == nj
        iv = params(3*nregion+4);
        IVmode = params(3*nregion+12);
        if IVmode == 1
            eq(iv1) = C(j,ii1) - iv; % specify current density (A/cm2)
        elseif IVmode == 2
            eq(iv1) = C(j,iv1) - iv; % specify cell potential (V)
        end
    end
% Equation 3: membrane phase current
    if j < bound(2)
        eq(ii2) = C(j,ii2);
    elseif j == bound(2)
        eq(ii2) = fluxright(j,ii2,C,Cp,params,pore2R,pore3);
    elseif j > bound(2) && j <= bound(5)
        eq(ii2) = fluxleft(j,ii2,C,Cp,params,pore2L,pore1)-...
            fluxright(j,ii2,C,Cp,params,pore2R,pore3);
    else
        eq(ii2) = C(j,ii2);
    end
% Equation 4: membrane phase potential
    if j < bound(2)
        eq(iv2) = C(j,iv2);
    elseif j >= bound(2) && j < bound(5)
        dx = mesh(j,C,iregion2,params);
        satR = 0.5*(pore2R(1)+pore3(1));
        V0 = 0.5*(calc_density(j,C)+calc_density(j+1,C));
        condmemvR = 0.5*(props2R(1)+props3(1));
        condmemlR = 0.5*(props2R(2)+props3(2));
        xivR = 0.5*(props2R(7)+props3(7));
        xilR = 0.5*(props2R(8)+props3(8));
        eq(iv2) = C(j,ii2) + (condmemlR*(C(j+1,iv2)-C(j,iv2))/dx +...
            condmemlR*xilR/F*0.1*V0*...
            (C(j+1,iplmem)-C(j,iplmem))/dx)*satR+...
            (condmemvR*xivR/F*(C(j+1,imuw)-C(j,imuw))/dx+...
            condmemvR*(C(j+1,iv2)-C(j,iv2))/dx)*(1-satR);
    elseif j == bound(5)
        eq(iv2) = fluxleft(j,ii2,C,Cp,params,pore2L,pore1);
    else
        eq(iv2) = C(j,iv2);
    end
% Equation 5: Water chemical potential in the membrane
    if j >= bound(2) && j <= bound(5)
        V0 = calc_density(j,C);
        eq(imuw) = C(j,imuw) - C(j,iplmem)*V0*0.1;
    else
        eq(imuw) = C(j,imuw);
    end
% Equation 6: Water flux in the membrane
    if j < bound(2)
        eq(iNwmem) = C(j,iNwmem);
    elseif j == bound(2)
        eq(iNwmem) = fluxright(j,iNwmem,C,Cp,params,pore2R,pore3);
    elseif j > bound(2) && j <= bound(5)
        eq(iNwmem) = fluxleft(j,iNwmem,C,Cp,params,pore2L,pore1)-...
            fluxright(j,iNwmem,C,Cp,params,pore2R,pore3);
    else
        eq(iNwmem) = C(j,iNwmem);
    end
% Equation 7: Oxygen flux
    if j == 1
        eq(iNO2) = C(j,iNO2);
    else
        eq(iNO2) = fluxleft(j,iNO2,C,Cp,params,pore2L,pore1)-...
            fluxright(j,iNO2,C,Cp,params,pore2R,pore3);
    end
% Equation 8: Nitrogen flux
    if j == 1
        ayN2avg = (ayN2in+ayN2out)/2;
        eq(iNN2) = C(j,iyN2)-ayN2avg;
    else
        eq(iNN2) = fluxleft(j,iNN2,C,Cp,params,pore2L,pore1)-...
            fluxright(j,iNN2,C,Cp,params,pore2R,pore3);
    end
% Equation 9: Water flux
    if j < bound(3)
        eq(iNw) = fluxleft(j,iNw,C,Cp,params,pore2L,pore1)-...
            fluxright(j,iNw,C,Cp,params,pore2R,pore3);
    elseif j == bound(3)
        eq(iNw) = fluxleft(j,iNw,C,Cp,params,pore2L,pore1);
    elseif j > bound(3) && j < bound(4)
        eq(iNw) = C(j,iNw);
    elseif j == bound(4)
        eq(iNw) = fluxright(j,iNw,C,Cp,params,pore2R,pore3);
    elseif j > bound(4)
        eq(iNw) = fluxleft(j,iNw,C,Cp,params,pore2L,pore1)-...
            fluxright(j,iNw,C,Cp,params,pore2R,pore3);
    end
% Equation 10: Hydrogen flux
    if j < nj
        eq(iNH2) = fluxleft(j,iNH2,C,Cp,params,pore2L,pore1)-...
            fluxright(j,iNH2,C,Cp,params,pore2R,pore3);
    else
        eq(iNH2) = C(j,iNH2);
    end
% Equation 11: Oxygen mole fraction
    if j == 1
        eq(iyO2) = C(j,iyO2);
    elseif j > 1 && j < bound(3)
        eq(iyO2) = stefan_maxwell(2,j,iyO2,C,params,pore2R,pore3);
    elseif j >= bound(3)  && j < bound(4)
        dx = mesh(j,C,iregion2,params);
        psiO2R = 0.5*(props2R(6)+props3(6));
        eq(iyO2) = C(j,iNO2) + (psiO2R*(C(j+1,iyO2)-C(j,iyO2)))/dx;
    elseif j >= bound(4)
        eq(iyO2) = C(j,iyO2)+C(j,iyN2)+C(j,iyw)+C(j,iyH2)-1;
    end
% Equation 12: Nitrogen mole fraction
    if j < bound(3)
        eq(iyN2) = stefan_maxwell(2,j,iyN2,C,params,pore2R,pore3);
    elseif j >= bound(3)  && j < bound(4)
        dx = mesh(j,C,iregion2,params);
        psiO2R = 0.5*(props2R(6)+props3(6));
        eq(iyN2) = C(j,iNN2) + (psiO2R*(C(j+1,iyN2)-C(j,iyN2)))/dx;
    elseif j >= bound(4)  && j < nj
        eq(iyN2) = stefan_maxwell(2,j,iyN2,C,params,pore2R,pore3);
    elseif j == nj
        cyN2avg = (cyN2in+cyN2out)/2;
        eq(iyN2) = C(j,iyN2)-cyN2avg;
    end
% Equation 13: Water mole fraction
    if j == 1
        eq(iyw) = aNwin - aNwDM - aNwout; % + aNwlin -aNwlDM - aNwlout;
    elseif j > 1 && j <= bound(3)
        eq(iyw) =  stefan_maxwell(1,j,iyw,C,params,pore2L,pore1);
    elseif j > bound(3)  && j < bound(4)
        eq(iyw) = C(j,iyw);
    elseif j >= bound(4)  && j < nj
        eq(iyw) = stefan_maxwell(2,j,iyw,C,params,pore2R,pore3);
    elseif j == nj
        eq(iyw) = cNwin+cNwDM-cNwout; %+cNwlin+cNwlDM-cNwlout;
    end
% Equation 14: Hydrogen mole fraction
    if j <= bound(3)
        eq(iyH2) = (C(j,iyw)+C(j,iyH2)+C(j,iyN2)+C(j,iyO2))-1;
    elseif j > bound(3)  && j <= bound(4)
        dx = mesh(j,C,iregion1,params);
        psiH2L = 0.5*(props2L(5)+props1(5));
        eq(iyH2) = C(j,iNH2) + (psiH2L*(C(j,iyH2)-C(j-1,iyH2)))/dx;
    elseif j > bound(4)  && j < nj
        eq(iyH2) = stefan_maxwell(1,j,iyH2,C,params,pore2L,pore1);
    else
        eq(iyH2) = C(j,iyH2);
    end
```

```
% Equation 16: Pressure
    if j == 1
        p0 = params(3*nregion+5);       % pressure (bar)
        eq(ipg) = C(j,ipg) - p0;
    elseif j > 1 && j <= bound(3)
        prmgL = 0.5*(pore2L(4)+pore1(4));
        eq(ipg) = calc_pressure(1,iregion1,j,C,prmgL,params);
    elseif j > bound(3)  && j < bound(4)
        eq(ipg) = C(j,ipg);
    elseif j >= bound(4)  && j < nj
        prmgR = 0.5*(pore2R(4)+pore3(4));
        eq(ipg) = calc_pressure(2,iregion2,j,C,prmgR,params);
    elseif j == nj
        eq(ipg) = C(j,ipg) - p0;
    end

% Equation 15: Temperature (energy balance)
    if j == 1
        T0 = params(3*nregion+6);       % initial temperature (K)
        Tcool = T0;
        [Hgasin,Hgasout,Hwin,Hwout,DHevapout] =...
            calc_enthalpy(j,T0,C,[0;0;aywin;ayH2in]);
        aconvin = Hgasin*agasNin+Hwin*aNwlin;
        agas1D = 0;
        for i = iNO2:iNH2
            agas1D = agas1D + fluxright(j,i,C,Cp,params,pore2R,pore3);
        end
        aliq1D = fluxright(j,iNwl,C,Cp,params,pore2R,pore3);
        aconvout = Hgasout*(agasNout+agas1D)+Hwout*(aNwlout+aliq1D);
        agenohm = C(j,ii1)*C(j,ii1)*0.05/100;
        acond = aNcond*DHevapout;
        htcoeff = params(3*nregion+15); % heat transfer coeff (W/cm2 K)
        condR = 0.5*(props2R(11)+props3(11));
        eq(iT) = agenohm-aconvout+aconvin+acond-...
           htcoeff*(C(j,iT)-Tcool) - ...
           energyfluxright(j,C,Cp,condR,[],params);
    elseif j > 1 && j <= bound(3)
        satL = 0.5*(pore2L(1)+pore1(1));
        satR = 0.5*(pore2R(1)+pore3(1));
        condL = 0.5*(props2L(11)+props1(11));
        condR = 0.5*(props2R(11)+props3(11));
        condmemvL = 0.5*(props2L(1)+props1(1));
        condmemvR = 0.5*(props2R(1)+props3(1));
        condmemlL = 0.5*(props2L(2)+props1(2));
        condmemlR = 0.5*(props2R(2)+props3(2));
        condmemL = condmemvL*(1-satL)+condmemlL*satL;
        condmemR = condmemvR*(1-satR)+condmemlR*satR;
        Tr1 = C(j,iT)/647.4;
        DHevap1 = 52053*(1-Tr1)^(0.3199-0.212*Tr1+0.25795*Tr1^2);
        Tr2 = C(j+1,iT)/647.4;
        DHevap2 = 52053*(1-Tr2)^(0.3199-0.212*Tr2+0.25795*Tr2^2);
        DHevapR = 0.5*(DHevap1+DHevap2);
        evap = (C(j+1,iNw)-C(j,iNw))*DHevapR;
        eq(iT) = energyfluxleft(j,C,Cp,condL,condmemL,params)-evap-...
            energyfluxright(j,C,Cp,condR,condmemR,params);
    elseif j > bound(3)  && j < bound(4)
        satL = 0.5*(pore2L(1)+pore1(1));
        satR = 0.5*(pore2R(1)+pore3(1));
        condL = 0.5*(props2L(11)+props1(11));
        condR = 0.5*(props2R(11)+props3(11));
        condmemvL = 0.5*(props2L(1)+props1(1));
        condmemvR = 0.5*(props2R(1)+props3(1));
        condmemlL = 0.5*(props2L(2)+props1(2));
        condmemlR = 0.5*(props2R(2)+props3(2));
        condmemL = condmemvL*(1-satL)+condmemlL*satL;
        condmemR = condmemvR*(1-satR)+condmemlR*satR;
        eq(iT) = energyfluxleft(j,C,Cp,condL,condmemL,params)-...
            energyfluxright(j,C,Cp,condR,condmemR,params);
    elseif j >= bound(4)  && j < nj
        satL = 0.5*(pore2L(1)+pore1(1));
        satR = 0.5*(pore2R(1)+pore3(1));
        condL = 0.5*(props2L(11)+props1(11));
        condR = 0.5*(props2R(11)+props3(11));
        condmemvL = 0.5*(props2L(1)+props1(1));
        condmemvR = 0.5*(props2R(1)+props3(1));
        condmemlL = 0.5*(props2L(2)+props1(2));
        condmemlR = 0.5*(props2R(2)+props3(2));
        condmemL = condmemvL*(1-satL)+condmemlL*satL;
        condmemR = condmemvR*(1-satR)+condmemlR*satR;
        Tr1 = C(j,iT)/647.4;
        DHevap1 = 52053*(1-Tr1)^(0.3199-0.212*Tr1+0.25795*Tr1^2);
        Tr2 = C(j-1,iT)/647.4;
        DHevap2 = 52053*(1-Tr2)^(0.3199-0.212*Tr2+0.25795*Tr2^2);
        DHevapL = 0.5*(DHevap1+DHevap2);
        evap = (C(j,iNw)-C(j-1,iNw))*DHevapL;
        eq(iT) = energyfluxleft(j,C,Cp,condL,condmemL,params)-evap-...
            energyfluxright(j,C,Cp,condR,condmemR,params);
    elseif j == nj
        T0 = params(3*nregion+6);       % initial temperature (K)
        Tin = T0;
        Tcool = T0;
        [Hgasin,Hgasout,Hwin,Hwout,DHevapout] = ...
            calc_enthalpy(j,Tin,C,[cyO2in;cyN2in;cywin;0]);
        cconvin = Hgasin*cgasNin+Hwin*cNwlin;
        ccond = cNcond*DHevapout;
        cgas1D = 0;
        for i = iNO2:iNH2
            cgas1D = cgas1D + fluxleft(j,i,C,Cp,params,pore2L,pore1);
        end
        cliq1D = fluxleft(j,iNwl,C,Cp,params,pore2L,pore1);
        cconvout = Hgasout*(cgasNout-cgas1D)+Hwout*(cNwout-cliq1D);
        cgenohm = C(j,ii1)*C(j,ii1)*0.05/100;
        htcoeff = params(3*nregion+15); % heat transfer coeff (W/cm2 K)
        condL = 0.5*(props2L(11)+props1(11));
        eq(iT) = cgenohm-cconvout+cconvin+ccond-...
            htcoeff*(C(j,iT)-Tcool)+...
            energyfluxleft(j,C,Cp,condL,[],params);
    end

% Equation 17: Membrane Thickness
    if j > bound(3)  && j <= bound(4)
        eq(iL) = C(j,iL)-C(j-1,iL);
    elseif j == bound(3)
        eq(iL) = C(j,iL) - C(j,itau);
    else
        eq(iL) = C(j,iL);
    end

% Equation 18: Membrane Expansion Fraction
    if j >= bound(3) && j < bound(4)
        dx = L(3)/(bound(iregion2+1)-bound(iregion2));
        Lmem = 0.5*(props2R(9)+props3(9));
        eq(itau) = C(j+1,itau)-C(j,itau)+Lmem*dx;
    else
        eq(itau) = C(j,itau);
    end

% Equation 19: Liquid Water Pressure
    kL = 0.1;
    pthru = C(j,ipg)+0.02;
    if j == 1
        eq(ipl) = C(j,iNwl)+kL*(C(j,ipl)-pthru)*...
            (tanh(C(j,ipl)-pthru)+1);
    elseif j > 1 && j <= bound(3)
        dx = mesh(j,C,iregion1,params);
        prmwL = 0.5*(pore2L(3)+pore1(3));
        visH2OL = (2695.3-6.6*0.5*(C(j,iT)+C(j-1,iT)))*1e-11;
        V0L = 0.5*(calc_density(j,C)+calc_density(j-1,C));
        eq(ipl) = C(j,iNwl)+prmwL/visH2OL*(C(j,ipl)-C(j-1,ipl))/dx/V0L;
    elseif j > bound(3) && j < bound(4)
        eq(ipl) = C(j,ipl);
    elseif j >= bound(4) && j < nj
        dx = mesh(j,C,iregion2,params);
        prmwR = 0.5*(pore2R(3)+pore3(3));
        visH2OR = 0.5*(2695.3-6.6*0.5*(C(j,iT)+C(j+1,iT)))*1e-11;
        V0R = 0.5*(calc_density(j,C)+calc_density(j+1,C));
        eq(ipl) = C(j,iNwl)+prmwR/visH2OR*(C(j+1,ipl)-C(j,ipl))/dx/V0R;
    elseif j == nj
        eq(ipl) = C(j,iNwl)+kL*(C(j,ipl)-pthru)*...
            (tanh(C(j,ipl)-pthru)+1);
    end

% Equation 20: Liquid Water Pressure in the membrane
    if j < bound(2)
        eq(iplmem) = C(j,iplmem);
    elseif j >= bound(2) && j < bound(5)
        dx = mesh(j,C,iregion2,params);
        V0 = 0.5*(calc_density(j,C)+calc_density(j+1,C));
        satR = 0.5*(pore2R(1)+pore3(1));
        condmemvR = 0.5*(props2R(1)+props3(1));
        condmemlR = 0.5*(props2R(2)+props3(2));
        alphavR = 0.5*(props2R(3)+props3(3));
        alphalR = 0.5*(props2R(4)+props3(4));
        xivR = 0.5*(props2R(7)+props3(7));
        xilR = 0.5*(props2R(8)+props3(8));
                eq(iplmem) = C(j,iNwmem) + satR*((condmemlR*xilR/F)*...
            (C(j+1,iv2)-C(j,iv2))/dx+(alphalR+condmemlR*xilR^2/F^2)*...
            0.1*V0*(C(j+1,iplmem)-C(j,iplmem))/dx) +...
            (1-satR)*((condmemvR*xivR/F)*(C(j+1,iv2)-C(j,iv2))/dx +...
            (alphavR+ condmemvR*xivR^2/F^2)*(C(j+1,imuw)-C(j,imuw))/dx);
    elseif j == bound(5)
```

```
        eq(iplmem) = fluxleft(j,iNwmem,C,Cp,params,pore2L,pore1);
    else
        eq(iplmem) = C(j,iplmem);
    end

% Equation 21: Liquid-water flux in porous media
    if j <= bound(3)
        eq(iNwl) = fluxleft(j,iNwl,C,Cp,params,pore2L,pore1)-...
            fluxright(j,iNwl,C,Cp,params,pore2R,pore3);
    elseif j > bound(3) && j < bound(4)
        eq(iNwl) = C(j,iNwl);
    elseif j >= bound(4)
        eq(iNwl) = fluxleft(j,iNwl,C,Cp,params,pore2L,pore1)-...
            fluxright(j,iNwl,C,Cp,params,pore2R,pore3);
    end

% Equation 22: Fluoride Flux
    if j < 0.5*(bound(4)+bound(5))
        eq(iNF) = fluxleft(j,iNF,C,Cp,params,pore2L,pore1) -...
            fluxright(j,iNF,C,Cp,params,pore2R,pore3);
    elseif j == 0.5*(bound(4)+bound(5))
        eq(iNF) = C(j,iNF);
    else
        eq(iNF) = fluxleft(j,iNF,C,Cp,params,pore2L,pore1) -...
            fluxright(j,iNF,C,Cp,params,pore2R,pore3);
    end

% Equation 23: Fluoride Ion Concentration
    DI = 2e-10;
    DGDL = 4.2e-9;
    if j < bound(2)
        dx = mesh(j,C,iregion2,params);
        eq(icF) = (C(j+1,icF)-C(j,icF))/dx +...
            (C(j,icF)+C(j+1,icF))*(C(j+1,iNF)+C(j,iNF))/(2*DGDL);
    elseif j >= bound(2) && j < 0.5*(bound(4)+bound(5))
        dx = mesh(j,C,iregion2,params);
        eq(icF) = (C(j+1,icF)-C(j,icF))/dx +...
            (C(j,icF)+C(j+1,icF))*(C(j+1,iNF)+C(j,iNF))/(2*DI);
    elseif j == 0.5*(bound(4)+bound(5))
        eq(icF) = (C(j+1,icF)-C(j,icF))/dx +...
            (C(j,icF)+C(j+1,icF))*(C(j+1,iNF)+C(j,iNF))/(2*DI)+...
            (C(j,icF)-C(j-1,icF))/dx +...
            (C(j,icF)+C(j-1,icF))*(C(j-1,iNF)+C(j,iNF))/(2*DI);
    elseif j > 0.5*(bound(4)+bound(5)) && j < bound(5)
        dx = mesh(j,C,iregion1,params);
        eq(icF) =  (C(j,icF)-C(j-1,icF))/dx +...
            (C(j,icF)+C(j-1,icF))*(C(j-1,iNF)+C(j,iNF))/(2*DI);
    else
        dx = mesh(j,C,iregion1,params);
        eq(icF) = (C(j,icF)-C(j-1,icF))/dx +...
            (C(j-1,icF)+C(j,icF))*(C(j,iNF)+C(j-1,iNF))/(2*DGDL);
    end
end
```

```matlab
function [pore1,props1,pore2L,props2L,pore2R,props2R,...
    pore3,props3] = calc_props(j,C,params)

nregion = params(1);
bound = params(2*nregion+3:3*nregion+3);
nj = bound(6);

if j > 1
    iregion = region(1,j,params);
    pore1 = calc_sat(iregion,j-1,C,params);
    props1 = calc_mem(iregion,j-1,C,pore1,params);
    pore2L = calc_sat(iregion,j,C,params);
    props2L = calc_mem(iregion,j,C,pore2L,params);
else
    iregion = region(1,j,params);
    pore2L = calc_sat(iregion,j,C,params);
    props2L = calc_mem(iregion,j,C,pore2L,params);
    pore1 = pore2L;
    props1 = props2L;
end

if j < nj
    iregion = region(2,j,params);
    pore3 = calc_sat(iregion,j+1,C,params);
    props3 = calc_mem(iregion,j+1,C,pore3,params);
    pore2R = calc_sat(iregion,j,C,params);
    props2R = calc_mem(iregion,j,C,pore2R,params);
else
    iregion = region(2,j,params);
    pore2R = calc_sat(iregion,j,C,params);
    props2R = calc_mem(iregion,j,C,pore2R,params);
    pore3 = pore2R;
    props3 = props2R;
end

end
```

```
function props = calc_mem(iregion,j,C,pore,params)
imuw = 5; iT = 15;
nregion = params(1);
EW = params(3*nregion+11); % membrane equivalent weight (g/mol)
sigma = params(3*nregion+10); % bulk-phase conductivity (S/cm)
rho_m = params(3*nregion+16);

MW0 = 18.0152; % molecular weight of water (g/mol)
R = 8.314; % ideal gas constant (J/mol K)
Tref = 30+273.15; % reference temperature (K)
fperc = 0.06; % conductivity percolation threshold
epsM = [0.0 0.3 1 0.3 0.0]; % membrane volume fractions
eps0 = [0.6 0.3 0 0.3 0.6]; % void fractions for gas transport
fwet = [0.6 0.3 0.0 0.3 0.6]; % fraction of hydrophilic pores
thcond = [0.015 0.003 0.0025 0.003 0.0125]; % eff thermal cond (W/cm K)
eta = [1.7 4.0 0.0 4.0 1.7]; % teflon loading

% membrane properties
visH2O = (2695.3-6.6*C(j,iT))*1e-11; % water viscosity
V0 = calc_density(j,C);
Vm = EW/rho_m;    % molar volume of the membrane (cm3/mol)
if C(j,imuw) > 0
    a = 1;
else
    a = exp(C(j,imuw)/(R*C(j,iT)));
end
xlmaxl = 22; % maximum water content
% calculate water content from isotherm (Weber & Newman 2004)
b3 = 36; b2 = -42.8; b1 = 20.45; b0 = 0.05;
xlamv = b3*a^3+b2*a^2+b1*a+b0;
sat = pore(1);
xlam = xlamv*(1-sat)+xlmaxl*sat;
% calculated membrane properties
fwater = (xlam*V0)/(Vm+xlam*V0); % volume fraction of water in the
membrane
cwater = xlam/(Vm+xlam*V0);
cwaterv = xlamv/(V0*xlamv+Vm);
cmemv = 1/(V0*xlamv+Vm);
fwaterv = xlamv*V0/(xlamv*V0+Vm);
fwaterl = xlmaxl*V0/(xlmaxl*V0+Vm);
% electroosmotic coefficient (mol H2O/mol H+)
if xlam < 1
    xiv = xlam;
else
    xiv = 1;
end
xil = 2.55*exp(4000/R*(1/Tref-1/C(j,iT)));

% mass transport coefficient (mol^2/(J cm s))
prmw = 1.8e-14;
prmw = prmw*(fwater/fwaterl)^2;
alphal = 1/V0*prmw/visH2O/0.1/V0;
DH2Om = 1.8e-5*exp(20000/R*(1/Tref-1/C(j,iT)));
DH2O = DH2Om*fwaterv;
xwaterv = cwaterv/(cwaterv+cmemv);
alphav = cwaterv*DH2O/R/C(j,iT)/(1-xwaterv);

```

```
% membrane conductivity (S/cm)
if fwaterv < fperc
    condmemv = 1e-5;
    disp('failure: membrane conductivity is zero')
elseif fwaterv >= 0.45
    sigp = 0.5*(0.45-fperc)^1.5;
    sigmxv = exp(15000/R*(1/Tref-1/C(j,iT)));
    condmemv = sigp*sigmxv;
else
    sigp = 0.5*(fwaterv-fperc)^1.5;
    sigmxv = exp(15000/R*(1/Tref-1/C(j,iT)));
    condmemv = sigp*sigmxv;
end
if fwaterl < fperc
    condmeml = 1e-5;
    disp('failure: membrane conductivity is zero')
elseif fwaterl >= 0.45
    sigp = 0.5*(0.45-fperc)^1.5;
    sigmxv = exp(15000/R*(1/Tref-1/C(j,iT)));
    condmeml = sigp*sigmxv;
else
    sigp = 0.5*(fwaterl-fperc)^1.5;
    sigmxv = exp(15000/R*(1/Tref-1/C(j,iT)));
    condmeml = sigp*sigmxv;
end
thickm = 1+V0*xlam/Vm*0.36;
% H2 permeation coefficient (mol/bar/cm/s)
psiH2l = 1.8e-11*exp(21000/R*(1/Tref-1/C(j,iT)));
psiH2v = (2.2e-11*fwaterv+2.9e-12)*exp(18000/R*...
    (1/Tref-1/C(j,iT)));
psiH2 = psiH2l*sat+(1-sat)*psiH2v;
% O2 permeation coefficient (mol/bar/cm/s)
psiO2l = 1.2e-11*exp(10000/R*(1/Tref-1/C(j,iT)));
psiO2v = (1.9e-11*fwaterv+1.1e-12)*exp(22000/R*...
    (1/Tref-1/C(j,iT)));
psiO2 = psiO2l*sat+(1-sat)*psiO2v;
% effective properties
condmemv = condmemv*epsM(iregion)^1.5;
condmeml = condmeml*epsM(iregion)^1.5;
alphav = alphav*epsM(iregion)^1.5;
alphal = alphal*epsM(iregion)^1.5;
psiH2 = psiH2*epsM(iregion)^1.5;
psiO2 = psiO2*epsM(iregion)^1.5;
xiv = xiv*epsM(iregion)^1.5;
xil = xil*epsM(iregion)^1.5;

porsolid = 1-eps0(iregion)-epsM(iregion);
porcarbon = porsolid*fwet(iregion);
tcond = thcond(iregion)*porcarbon^1.5;
cond = sigma*(porcarbon*eta(iregion))^1.5;

props = [condmemv,condmeml,alphav,alphal,psiH2,psiO2,xiv,xil,...
    thickm,tcond,cond,V0];
end
```

```
function pore = calc_sat(iregion,j,C,params)

iT = 15; ipg = 16; ipl = 19;

nregion = params(1);
eps0 = [0.6 0.3 0 0.3 0.6];  % void fractions for gas transport
fwet = [0.5 0.3 0.0 0.3 0.5];        % fraction of hydrophilic pores
anghl = [45 80 90.02 80 45];       % hydrophilic contact angle (degrees)
angho = [110 100 90.02 100 110];  % hydrophobic contact angle (degrees)
rad1 = [6 0.2 0.00125 0.2 6]; % characteristic pore size dist 1 (um)
rad2 = [0.7 0.05 0.00125 0.05 0.7];% characteristic pore size dist 2(um)
wide1 = [0.6 1.2 0.3 1.2 0.6];     % pore size distribution 1 width
wide2 = [0.6 0.5 0.3 0.5 0.6];     % pore size distribution 2 width
fr1 = [1 0.5 1 0.5 1];             % fraction of pore size distribution 1
perm = [0.6e-12 8e-15 1.8e-14 8e-15 0.6e-12]; %abs permeability (cm2)

% calculate wetting phase percolation threshold, satw0
if iregion == 3
    satw0 = 0;
else
    satw0 = -5.2363*eps0(iregion)^5+17.075*eps0(iregion)^4-...
        21.717*eps0(iregion)^3+13.696*eps0(iregion)^2-...
        4.8164*eps0(iregion)+0.9989;
end

% calculate nonwetting irreducible saturation (maximum saturation),
satnw0
satnw0 = 1-satw0;
if satnw0 < 0.85
    satnw0 = 0.85;
end

% capillary pressure definition
pc = C(j,ipl)-C(j,ipg); % bar
surft = (123.98-0.17393*C(j,iT))*1e-3; % surface tension (N/cm)

%%
if fwet(iregion) < 0.15 || fwet(iregion) > 0.85
    % assume single pore size distribution
    % composite angle
    angc = 180/pi*acos(fwet(iregion)*cos(anghl(iregion)*pi/180)+...
        (1-fwet(iregion))*cos(angho(iregion)*pi/180));
    if angc < 90 % hydrophilic
        if pc < 0
            % critical radius (um)
            r= -2*surft*cos(angc)/pc*10;
            % calculate differential pore volume
            x1 = log(r/rad1(iregion))/(wide1(iregion)*sqrt(2));
            x2 = log(r/rad2(iregion))/(wide2(iregion)*sqrt(2));
            vr1 = 0.5*(1+erf(x1));
            vr2 = 0.5*(1+erf(x2));
            vr = fr1(iregion)*vr1+(1-fr1(iregion))*vr2;
            pr1 = 0.5*(1+erf(x1-wide1(iregion)*sqrt(2)));
            pr2 = 0.5*(1+erf(x2-wide2(iregion)*sqrt(2)));
            pr = fr1(iregion)*pr1+(1-fr1(iregion))*pr2;
            if abs (1-erf(x1)) > 1e-13
                rkw1 = (1-erf(x1-wide1(iregion)/sqrt(2)))/(1-erf(x1));
                rk1 = rad1(iregion)*exp(wide1(iregion)^2/2)*rkw1;
            else
                rk1 = 0;
            end
            if abs (1-erf(x2)) > 1e-13
                rkw2 = (1-erf(x2-wide2(iregion)/sqrt(2)))/(1-erf(x2));
                rk2 = rad2(iregion)*exp(wide2(iregion)^2/2)*rkw2;
            else
                rk2 = 0;
            end
            sw = vr;
            prw = pr;
            prg = 1-pr;
        elseif pc >= 0 % all pores filled
            sw = 1;
            prw = 1;
            prg = 0;
            rk1 = 0;
            rk2 = 0;
        end
        sat = sw;
        if sat <= satw0
            effsat = 0;
            sateff = 0;
            epsg = eps0(iregion)*(1-satw0);
            taug = (eps0(iregion)*(satnw0-satw0))^-0.5;
        elseif sat >= satnw0
            taug = 1e4;
            epsg = 0;
            sat = satnw0;
            sateff = (satnw0-satw0)/(1-satw0);
            effsat = 1;
        else
            effsat = (sat-satw0)/(satnw0-satw0);
            sateff = (sat-satw0)/(1-satw0);
            epsg = eps0(iregion)*(1-sat);
            taug = (eps0(iregion)*(satnw0-sat))^-0.5;
        end
        rk = rk1*fr1(iregion)+rk2*(1-fr1(iregion));
        prmw = perm(iregion)*prw*sateff^2;
        prmg = perm(iregion)*prg*(1-sateff)^2;
    else % hydrophobic
        if pc <= 0 % all pores empty
            snw = 0;
            prw = 0;
            prg = 1;
            rk1 = rad1(iregion)*exp(wide1(iregion)^2/2);
            rk2 = rad2(iregion)*exp(wide2(iregion)^2/2);
        else % fill up HO pores
            r = -2*surft*cos(angc*pi/180)/pc*10;
            x1 = log(r/rad1(iregion))/(wide1(iregion)*sqrt(2));
            x2 = log(r/rad2(iregion))/(wide2(iregion)*sqrt(2));
            vr1 = 0.5*(1+erf(x1));
            vr2 = 0.5*(1+erf(x2));
            vr = fr1(iregion)*vr1+(1-fr1(iregion))*vr2;
            pr1 = 0.5*(1+erf(x1-wide1(iregion)*sqrt(2)));
            pr2 = 0.5*(1+erf(x2-wide2(iregion)*sqrt(2)));
```

```
                pr = fr1(iregion)*pr1+(1-fr1(iregion))*pr2;
                snw = 1-vr;
                pnw = 1-pr;
                pgnw = pr;
                prw = pnw;
                prg = pgnw;
                if abs(1+erf(x1)) > 1e-13
                    rk1 = (1+erf(x1-wide1(iregion)/sqrt(2)))/(1+erf(x1))*...
                        rad1(iregion)*exp(wide1(iregion)^2/2);
                else
                    rk1 = 0;
                end
                if abs(1+erf(x2)) > 1e-13
                    rk2 = (1+erf(x2-wide2(iregion)/sqrt(2)))/(1+erf(x2))*...
                        rad2(iregion)*exp(wide2(iregion)^2/2);
                else
                    rk2 = 0;
                end
            end
            sat = snw;
            rk = rk2*(1-fr1(iregion))+rk1*fr1(iregion);
            if sat <= satw0
                sateff = 0;
                effsat = 0;
                epsg = eps0(iregion)*(1-satw0);
                taug = (eps0(iregion)*(satnw0-satw0))^-0.5;
            elseif sat >= satnw0
                epsg = 0;
                taug = 1e4;
                rk = 0;
                sateff = (satnw0-satw0)/(1-satw0);
                effsat = 1;
                sat = satnw0;
            else
                effsat = (sat-satw0)/(satnw0-satw0);
                sateff = (sat-satw0)/(1-satw0);
                epsg = eps0(iregion)*(1-sat);
                taug = (eps0(iregion)*(satnw0-sat))^-0.5;
            end
            prmw = perm(iregion)*prw*sateff^2;
            prmg = perm(iregion)*(1-sateff)^2*prg;
        end
else
    % assume two separate pore networks
    % composite angles
    angchl = 180/pi*acos(0.85*cos(anghl(iregion)*pi/180)+...
        0.15*cos(angho(iregion)*pi/180));
    angcho = 180/pi*acos(0.15*cos(anghl(iregion)*pi/180)+...
        0.85*cos(angho(iregion)*pi/180));
    if pc < 0 % fill up HI pores
        r = -2*surft*cos(angchl*pi/180)/pc*10;
        x1 = log(r/rad1(iregion))/(wide1(iregion)*sqrt(2));
        x2 = log(r/rad2(iregion))/(wide2(iregion)*sqrt(2));
        vr1 = 0.5*(1+erf(x1));
        vr2 = 0.5*(1+erf(x2));
        vr = fr1(iregion)*vr1+(1-fr1(iregion))*vr2;
        pr1 = 0.5*(1+erf(x1-wide1(iregion)*sqrt(2)));
        pr2 = 0.5*(1+erf(x2-wide2(iregion)*sqrt(2)));
        pr = fr1(iregion)*pr1+(1-fr1(iregion))*pr2;
        sw = vr;
        snw = 0;
        pw = pr;
        pnw = 0;
        pgw = 1-pr;
        pgnw = 1;
        denom1 = 1-erf(x1);
        denom2 = 1-erf(x2);
        if abs(denom1) > 1e-13
            rkw1 = (1-erf(x1-wide1(iregion)^2/sqrt(2)))/denom1;
            rknw1 = 1;
            rk1 = rad1(iregion)*exp(wide1(iregion)^2/2)*...
                (fwet(iregion)*rkw1+(1-fwet(iregion))*rknw1);
        else
            rk1 = (1+erf(x1-wide1(iregion)/sqrt(2)))/(1+erf(x1))*...
                rad1(iregion)*exp(wide1(iregion)^2/2);
        end
        if abs(denom2) > 1e-13
            rkw2 = (1-erf(x2-wide2(iregion)^2/sqrt(2)))/denom2;
            rknw2 = 1;
            rk2 = rad2(iregion)*exp(wide2(iregion)^2/2)*...
                (fwet(iregion)*rkw2+(1-fwet(iregion))*rknw2);
        else
            rk2 = (1+erf(x2-wide2(iregion)/sqrt(2)))/(1+erf(x2))*...
                rad2(iregion)*exp(wide2(iregion)^2/2);
        end
    elseif pc == 0 % all HI filled, HO empty
        sw = 1;
        snw = 0;
        pw = 1;
        pnw = 0;
        pgw = 0;
        pgnw = 1;
        rk1 = rad1(iregion);
        rk2 = rad2(iregion);
    elseif pc > 0 % all HI filled, fill HO
        r = -2*surft*cos(angcho*pi/180)/pc*10;
        x1 = log(r/rad1(iregion))/(wide1(iregion)*sqrt(2));
        x2 = log(r/rad2(iregion))/(wide2(iregion)*sqrt(2));
        vr1 = 0.5*(1+erf(x1));
        vr2 = 0.5*(1+erf(x2));
        vr = fr1(iregion)*vr1+(1-fr1(iregion))*vr2;
        pr1 = 0.5*(1+erf(x1-wide1(iregion)*sqrt(2)));
        pr2 = 0.5*(1+erf(x2-wide2(iregion)*sqrt(2)));
        pr = fr1(iregion)*pr1+(1-fr1(iregion))*pr2;
        sw = 1;
        snw = 1-vr;
        pw = 1;
        pnw = 1-pr;
        pgw = 0;
        pgnw = pr;
        denom1 = 1+erf(x1);
        denom2 = 1+erf(x2);
        if abs(denom1) > 1e-13
            rk1 = (1+erf(x1-wide1(iregion)/sqrt(2)))/denom1*...
```

```
                rad1(iregion)*exp(wide1(iregion)^2/2);
        else
            rk1 = 0;
        end
        if abs(denom2) > 1e-13
            rk2 = (1+erf(x2-wide2(iregion)/sqrt(2)))/denom2*...
                rad2(iregion)*exp(wide2(iregion)^2/2);
        else
            rk2 = 0;
        end
    end
    sat = fwet(iregion)*sw+(1-fwet(iregion))*snw;
    rk = rk2*(1-fr1(iregion))+rk1*fr1(iregion);
    if sat <= satw0
        sateff = 0;
        effsat = 0;
        epsg = eps0(iregion)*(1-satw0);
        taug = (eps0(iregion)*(satnw0-satw0))^-0.5;
    elseif sat >= satnw0
        epsg = 0;
        taug = 1e4;
        rk = 0;
        sateff = (satnw0-satw0)/(1-satw0);
        effsat = 1;
        sat = satnw0;
    else
        sateff = (sat-satw0)/(1-satw0);
        effsat = (sat-satw0)/(satnw0-satw0);
        epsg = eps0(iregion)*(1-sat);
        taug = (eps0(iregion)*(satnw0-sat))^-0.5;
    end
    prw = (fwet(iregion)*pw+(1-fwet(iregion))*pnw);
    prg = (fwet(iregion)*pgw+(1-fwet(iregion))*pgnw);
    prmw = perm(iregion)*prw*sateff^2;
    prmg = perm(iregion)*prg*(1-sateff)^2;
end

%% set parameters if there is no water
if C(j,ipl) <= 0
    epsg = eps0(iregion);
    taug = eps0(iregion)^-0.5;
end
if iregion == 3
    prmw = perm(iregion);
end

prmg = prmg*1e6;
if prmg < 1e-14
    prmg = 1e-14;
end
pore = [sat,effsat,prmw,prmg,rk,epsg,taug];
end
```

```matlab
function energyfluxleft = energyfluxleft(j,C,Cp,sigma,kappa,params)
% Calculates the flux exiting the box to the left of point j
    iNwmem = 6; iNO2 = 7; iNN2 = 8; iNw = 9; iNH2 = 10;
    iyO2 = 11; iyH2 = 14; iT = 15; ipg = 16; iNwl = 21;

% parameters
nregion = params(1);
bound = params(2*nregion+3:3*nregion+3);
nj = bound(6);
F = 96485; % Faraday's constant (C/mol)
R = 83.14;    % ideal gas constant (cm3 bar/mol K)

iregion = region(1,j,params);
dx = mesh(j,C,iregion,params);
thcond = [0.015 0.003 0.0025 0.003 0.0125]; % eff thermal cond (W/cm K)

% Conduction
cond = -thcond(iregion)*(C(j,iT)-C(j-1,iT))/dx;

if j ~= 1
    a = [28.11         31.15       32.24      27.14     ];
    b = [-3.68e-6     -1.357e-2    1.924e-3   9.274e-3 ];
    c = [1.746e-5      2.68e-5     1.055e-5   -1.381e-5];
    d = [-1.065e-8    -1.168e-8   -3.596e-9   7.645e-9 ];
    Cpr = a + b.*C(j-1:j,iT) + c.*(C(j-1:j,iT)).^2 +d.*(C(j-1:j,iT)).^3;
    Cpgas = Cpr*C(j-1:j,iyO2:iyH2)';
    CpgasL = (Cpgas(1,1)+Cpgas(2,2))/2;
    Cpw = (2.7637e5-2090.1*C(j-1:j,iT)+8.125*C(j-1:j,iT).^2-...
        1.4116e-2*C(j-1:j,iT).^3+9.3701e-6*C(j-1:j,iT).^4)/1000;
    CpwL = 0.5*(Cpw(1)+Cpw(2));
end

% Convection (Flux into the box to the left)
if j > 1 && j <= bound(3)
    gasflux = C(j,iNH2)+C(j,iNw)+C(j-1,iNO2)+C(j-1,iNN2);
elseif j > bound(4)  && j <= nj
    gasflux = C(j-1,iNO2)+C(j-1,iNN2)+C(j-1,iNw)+C(j,iNH2);
else
    gasflux = 0;
end

if j ~= 1
    T = ((C(j,iT)+C(j-1,iT))/2);
end

if j == 1
    conv = 0;
elseif j > 1  && j <= bound(2)
    conv = (CpgasL*gasflux+CpwL*C(j,iNwl))*T;
elseif j > bound(2)  && j <= bound(3)
    conv = (CpgasL*gasflux+CpwL*C(j,iNwl)+CpwL*C(j-1,iNwmem))*T;
elseif j > bound(3)  && j <= bound(4)
    conv = CpwL*C(j-1,iNwmem)*T;
elseif j > bound(4)  && j <= bound(5)
    conv = (CpgasL*gasflux+CpwL*C(j-1,iNwl)+CpwL*C(j-1,iNwmem))*T;
elseif j > bound(5)  && j < nj
    conv = (CpgasL*gasflux+CpwL*C(j-1,iNwl))*T;
```

```matlab
elseif j == nj
    conv = 0;
end

% Reaction terms
% reaction 1 = HOR heat generation
% reaction 2 = ORR heat generation
% reaction 3 = ohmic heating
st = [1 2*F -4*F];

heat = heat_react(j,iregion,C,sigma,kappa);
if j ~= 1
    heatL = heat_react(j-1,iregion,C,sigma,kappa);
else
    heatL = heat;
end
w = 0.75;
acc = 0;

gen = st*(w*heat+(1-w)*heatL)*dx/2;

if isempty(Cp) == 0 % transient
    dt = params(3*nregion+19); % time spacing
    a = [28.11         31.15       32.24      27.14    ];
    b = [-3.68e-6     -1.357e-2    1.924e-3   9.274e-3 ];
    c = [1.746e-5      2.68e-5     1.055e-5   -1.381e-5];
    d = [-1.065e-8    -1.168e-8   -3.596e-9   7.645e-9 ];
    Cprp = a + b.*Cp(j-1:j,iT) + c.*(Cp(j-1:j,iT)).^2 +...
        d.*(Cp(j-1:j,iT)).^3;
    Cpgasp = Cp(j-1:j,iyO2:iyH2)*Cprp';
    CpgasLp = (Cpgasp(1,1)+Cpgasp(2,2))/2;
    CT = (C(j,ipg)/C(j,iT)+C(j-1,ipg)/C(j-1,iT))/(2*R);
    CTp = (Cp(j,ipg)/Cp(j,iT)+Cp(j-1,ipg)/Cp(j-1,iT))/(2*R);
    dTdt = w*0.5*(CpgasL*CT*C(j,iT)-CpgasLp*CTp*Cp(j,iT))/dt+...
         (1-w)*0.5*(CpgasL*CT*C(j-1,iT)-CpgasLp*CTp*Cp(j-1,iT))/dt;
    acc = dTdt*dx/2;
end

% Flux leaving the box to the left
energyfluxleft = cond + conv + gen - acc;

end
```

```
function energyfluxright = energyfluxright(j,C,Cp,sigma,kappa,params)
% Calculates the flux exiting the box to the left of point j
    iNwmem = 6; iNO2 = 7; iNN2 = 8; iNw = 9; iNH2 = 10;
    iyO2 = 11; iyH2 = 14; iT = 15; ipg = 16; iNwl = 21;

% parameters
nregion = params(1);
bound = params(2*nregion+3:3*nregion+3);
nj = bound(6);
F = 96485; % Faraday's constant (C/mol)
R = 83.14;    % ideal gas constant (cm3 bar/mol K)

iregion = region(2,j,params);
dx = mesh(j,C,iregion,params);

thcond = [0.015 0.003 0.0025 0.003 0.0125];
% Conduction
cond = -thcond(iregion)*(C(j+1,iT)-C(j,iT))/dx;

if j ~= nj
    a = [28.11         31.15       32.24      27.14    ];
    b = [-3.68e-6     -1.357e-2    1.924e-3   9.274e-3 ];
    c = [1.746e-5      2.68e-5     1.055e-5   -1.381e-5];
    d = [-1.065e-8    -1.168e-8   -3.596e-9   7.645e-9 ];
    Cpr = a + b.*C(j:j+1,iT) + c.*C(j:j+1,iT).^2 +d.*C(j:j+1,iT).^3;
    Cpgas = Cpr*C(j:j+1,iyO2:iyH2)';
    CpgasR = (Cpgas(1,1)+Cpgas(2,2))/2;
    Cpw = (2.7637e5-2090.1*C(j:j+1,iT)+8.125*C(j:j+1,iT).^2-...
        1.4116e-2*C(j:j+1,iT).^3+9.3701e-6*C(j:j+1,iT).^4)/1000;
    CpwR = 0.5*(Cpw(1)+Cpw(2));
end

% Convection (Flux into the box to the left)
if j >= 1 && j < bound(3)
    gasflux = C(j+1,iNH2)+C(j+1,iNw)+C(j,iNO2)+C(j,iNN2);
elseif j >= bound(4) && j < nj
    gasflux = C(j,iNO2)+C(j,iNN2)+C(j,iNw)+C(j+1,iNH2);
else
    gasflux = 0;
end

if j ~= nj
    T = ((C(j,iT)+C(j+1,iT))/2);
end

if j == 1
    conv = 0;
elseif j > 1 && j <= bound(2)
    conv = (CpgasR*gasflux+CpwR*C(j+1,iNwl))*T;
elseif j > bound(2) && j < bound(3)
    conv = (CpgasR*gasflux+CpwR*C(j+1,iNwl)+CpwR*C(j,iNwmem))*T;
elseif j >= bound(3) && j < bound(4)
    conv = CpwR*C(j,iNwmem)*T;
elseif j >= bound(4) && j < bound(5)
    conv = (CpgasR*gasflux+CpwR*C(j,iNwl)+CpwR*C(j,iNwmem))*T;
elseif j >= bound(5) && j < nj
    conv = (CpgasR*gasflux+CpwR*C(j,iNwl))*T;
```

```
elseif j == nj
    conv = 0;
end

% Reaction terms
% reaction 1 = HOR heat generation
% reaction 2 = ORR heat generation
% reaction 3 = ohmic heating
st = [1 2*F -4*F];

heat = heat_react(j,iregion,C,sigma,kappa);
if j ~= 1
    heatR = heat_react(j+1,iregion,C,sigma,kappa);
else
    heatR = heat;
end

w = 0.75;
acc = 0;

gen = st*(w*heat+(1-w)*heatR)*dx/2;

if isempty(Cp) == 0 % transient
    dt = params(3*nregion+19); % time spacing
    a = [28.11         31.15       32.24      27.14     ];
    b = [-3.68e-6     -1.357e-2    1.924e-3   9.274e-3 ];
    c = [1.746e-5      2.68e-5     1.055e-5   -1.381e-5];
    d = [-1.065e-8    -1.168e-8   -3.596e-9   7.645e-9 ];
    CT = (C(j,ipg)/C(j,iT)+C(j+1,ipg)/C(j+1,iT))/(2*R);
    CTp = (Cp(j,ipg)/Cp(j,iT)+Cp(j+1,ipg)/Cp(j+1,iT))/(2*R);
    Cprp = a + b.*Cp(j:j+1,iT) + c.*(Cp(j:j+1,iT)).^2 +...
        d.*(Cp(j:j+1,iT)).^3;
    Cpgasp = Cp(j:j+1,iyO2:iyH2)*Cprp';
    CpgasRp = (Cpgasp(1,1)+Cpgasp(2,2))/2;
    dTdt = w*0.5*(CT*CpgasR*C(j,iT)-CTp*CpgasRp*Cp(j,iT))/dt + ...
        (1-w)*0.5*(CT*CpgasR*C(j+1,iT)-CTp*CpgasRp*Cp(j+1,iT))/dt;
    acc = dTdt*dx/2;
end

% Flux leaving the box to the left
energyfluxright = cond + conv - gen + acc;
end
```

```
function NL = fluxleft(j,i,C,Cp,params,pore2L,pore1)
% Calculates the flux exiting the box to the left of point j
iv1 = 2; ii2 = 3; iv2 = 4; iNwmem = 6; iNO2 = 7; iNN2 = 8;
iNw = 9; iNH2 = 10; iT = 15; ipg = 16; iNwl = 21; iNF = 22; icF = 23;

% parameters
nregion = params(1);
nspecies = params(2);
bound = params(2*nregion+3:3*nregion+3);
R = 83.14;     % ideal gas constant (cm3 bar/mol K)
F = 96485;     % Faraday's constant (C/mol)

iregion = region(1,j,params);
dx = mesh(j,C,iregion,params);

epsM = [0.0 0.3 1 0.3 0.0]; % membrane volume fractions
eps0 = [0.6 0.3 0 0.3 0.6]; % void fractions for gas transport

% Flux in the box to the left
if i == iNH2
    flux = C(j,i);
elseif i == iNw
    if j <= bound(3)
        flux = C(j,i);
    else
        flux = C(j-1,i);
    end
elseif i == iNwl
    if j <= bound(3)
        flux = C(j,i);
    else
        flux = C(j-1,i);
    end
elseif i == iNF
    if j <= 0.5*(bound(4)+bound(5))
        flux = C(j,i);
    elseif j > 0.5*(bound(4)+bound(5))
        flux = C(j-1,i);
    end
else
    flux = C(j-1,i);
end

w = 0.75;
% Reaction terms
% reaction 1 = HOR
% reaction 2 = ORR
% reaction 3 = water transfer from membrane to gas
% reaction 4 = water transfer from membrane to liquid
% reaction 5 = water transfer from liquid to gas
% reaction 6 = H2O2 and fluoride generation
st = zeros(23,6);
st(iNO2,:) = [0 -1 0 0 0 -1];
st(iNwmem,:) = [0 2 -1 -1 0 0];
st(iNH2,:) = [-1 0 0 0 0 0];
st(iNw,:) = [0 0 1 0 1 0];
st(ii2,:) = [2*F -4*F 0 0 0 -2*F];
st(iNwl,:) = [0 0 0 1 -1 0];
st(iNF,:) = [0 0 0 0 0 1/dx];
rate = react(j,C,iregion,pore2L);
if j == 1
    rateL = rate;
else
    rateL = react(j-1,C,iregion,pore1);
end
gen = st(i,:)*(w*rate+(1-w)*rateL)*dx/2;

acc = 0;
if isempty(Cp) == 0  %transient
    dt = params(3*nregion+19); % time spacing
    if i == ii2
        acc = 0;
    elseif i == iNwmem
        cwaterv = calc_cwater(j,iregion,C,params);
        cwatervp = calc_cwater(j,iregion,Cp,params);
        if j > 1
            cwatervL = calc_cwater(j-1,iregion,C,params);
            cwatervpL = calc_cwater(j-1,iregion,Cp,params);
        else
            cwatervL = cwaterv;
            cwatervpL = cwatervp;
        end
        dcdt = w*0.5*(cwaterv-cwatervp)/dt+...
            (1-w)*0.5*(cwatervL-cwatervpL)/dt;
        acc = epsM(iregion)*dcdt*dx/2;
    elseif i == iNwl
        V0 = calc_density(j,C);
        pore = calc_sat(iregion,j,C,params);
        V0p = calc_density(j,Cp);
        porep = calc_sat(iregion,j,Cp,params);
        if j == 1
            V0L = V0;
            poreL = pore;
            V0Lp = V0p;
            poreLp = porep;
        else
            V0L = calc_density(j-1,C);
            poreL = calc_sat(iregion,j-1,C,params);
            V0Lp = calc_density(j-1,Cp);
            poreLp = calc_sat(iregion,j-1,Cp,params);
        end
        dcdt = w*0.5*(pore(1)/V0-porep(1)/V0p)/dt+...
            (1-w)*0.5*(poreL(1)/V0L-poreLp(1)/V0Lp)/dt;
        acc = eps0(iregion)*dcdt*dx/2;
    elseif i == iNF
        if j == 1
            dcdt = (C(j,icF)-Cp(j,icF))/dt;
        else
            dcdt = w*0.5*(C(j,icF)-Cp(j,icF))/dt+...
                (1-w)*0.5*(C(j-1,icF)-Cp(j-1,icF))/dt;
        end
        acc = (1-eps0(iregion))*dcdt*dx/2;
    else
        if j == 1
```

```
            pore = calc_sat(iregion,j,C,params);
            porep = calc_sat(iregion,j,Cp,params);
            CT = C(j,ipg)/(C(j,iT)*R);
            CTp = Cp(j,ipg)/(Cp(j,iT)*R);
            dcdt = (CT*(1-pore(1))*C(j,i+nspecies)-...
                (1-porep(1))*CTp*Cp(j,i+nspecies))/dt;
        else
            pore = calc_sat(iregion,j,C,params);
            poreL = calc_sat(iregion,j-1,C,params);
            porep = calc_sat(iregion,j,Cp,params);
            porepL = calc_sat(iregion,j-1,Cp,params);
            CT = (C(j,ipg)/C(j,iT)+C(j-1,ipg)/C(j-1,iT))/(2*R);
            CTp = (Cp(j,ipg)/Cp(j,iT)+Cp(j-1,ipg)/Cp(j-1,iT))/(2*R);
            dcdt = w*0.5*(CT*(1-pore(1))*C(j,i+nspecies)-...
                CTp*(1-porep(1))*Cp(j,i+nspecies))/dt+...
                (1-w)*0.5*(CT*(1-poreL(1))*C(j-1,i+nspecies)-...
                CTp*(1-porepL(1))*Cp(j-1,i+nspecies))/dt;
        end
        acc = eps0(iregion)*dcdt*dx/2;
    end
end

NL = flux + gen - acc;
end
```

```
function NR = fluxright(j,i,C,Cp,params,pore2R,pore3)
% Calculates the flux exiting the box to the right of point j
iv1 = 2; ii2 = 3; iv2 = 4; iNwmem = 6; iNO2 = 7; iNN2 = 8;
iNw = 9; iNH2 = 10; iT = 15; ipg = 16; iNwl = 21; iNF = 22; icF = 23;

% parameters
nregion = params(1);
nspecies = params(2);
bound = params(2*nregion+3:3*nregion+3);
nj = bound(6);
R = 83.14;    % ideal gas constant (cm3 bar/mol K)
F = 96485;                      % Faraday's constant (C/mol)

iregion = region(2,j,params);
dx = mesh(j,C,iregion,params);
epsM = [0.0 0.3 1 0.3 0.0]; % membrane volume fractions
eps0 = [0.6 0.3 0 0.3 0.6]; % void fractions for gas transport;

% Flux in the box to the right
if i == iNH2
    flux = C(j+1,i);
elseif i == iNw
    if j <= bound(3)
        flux = C(j+1,i);
    else
        flux = C(j,i);
    end
elseif i == iNwl
    if j <= bound(3)
        flux = C(j+1,i);
    else
        flux = C(j,i);
    end
elseif i == iNF
    if j <= 0.5*(bound(4)+bound(5))
        flux = C(j+1,i);
    elseif j > 0.5*(bound(4)+bound(5))
        flux = C(j,i);
    end
else
    flux = C(j,i);
end

w = 0.75;
% Reaction terms
% reaction 1 = HOR
% reaction 2 = ORR
% reaction 3 = water transfer from membrane to gas
% reaction 4 = water transfer from membrane to liquid
% reaction 5 = water transfer from liquid to gas
% reaction 6 = H2O2 and fluoride generation
st = zeros(23,6);
st(iNO2,:) = [0 -1 0 0 0 -1];
st(iNwmem,:) = [0 2 -1 -1 0 0];
st(iNH2,:) = [-1 0 0 0 0 0];
st(iNw,:) = [0 0 1 0 1 0];
st(ii2,:) = [2*F -4*F 0 0 0 -2*F];
st(iNwl,:) = [0 0 0 1 -1 0];
st(iNF,:) = [0 0 0 0 0 1/dx];
rate = react(j,C,iregion,pore2R);
if j == nj
    rateR = rate;
else
    rateR = react(j+1,C,iregion,pore3);
end
gen = st(i,:)*(w*rate+(1-w)*rateR)*dx/2;

acc = 0;
if isempty(Cp) == 0 % transient
    dt = params(3*nregion+19); % time spacing
    if i == ii2
        acc = 0;
    elseif i == iNwmem
        cwaterv = calc_cwater(j,iregion,C,params);
        cwatervp = calc_cwater(j,iregion,Cp,params);
        if j < nj
            cwatervR = calc_cwater(j+1,iregion,C,params);
            cwatervpR = calc_cwater(j+1,iregion,Cp,params);
        else
            cwatervR = cwaterv;
            cwatervpR = cwatervp;
        end
        dcdt = w*0.5*(cwaterv-cwatervp)/dt+...
            (1-w)*0.5*(cwatervR-cwatervpR)/dt;
        acc = epsM(iregion)*dcdt*dx/2;
    elseif i == iNwl
        V0 = calc_density(j,C);
        pore = calc_sat(iregion,j,C,params);
        V0p = calc_density(j,Cp);
        porep = calc_sat(iregion,j,Cp,params);
        if j == nj
            V0R = V0;
            poreR = pore;
            V0Rp = V0p;
            poreRp = porep;
        else
            V0R = calc_density(j+1,C);
            poreR = calc_sat(iregion,j+1,C,params);
            V0Rp = calc_density(j+1,Cp);
            poreRp = calc_sat(iregion,j+1,Cp,params);
        end
        dcdt = w*0.5*(pore(1)/V0-porep(1)/V0p)/dt+...
            (1-w)*0.5*(poreR(1)/V0R-poreRp(1)/V0Rp)/dt;
        acc = eps0(iregion)*dcdt*dx/2;
    elseif i == iNF
        if j == nj
            dcdt = (C(j,icF)-Cp(j,icF))/dt;
        else
            dcdt = w*0.5*(C(j,icF)-Cp(j,icF))/dt+...
                (1-w)*0.5*(C(j+1,icF)-Cp(j+1,icF))/dt;
        end
        acc = (1-eps0(iregion))*dcdt*dx/2;
    else
        if j == nj
```

```
            pore = calc_sat(iregion,j,C,params);
            porep = calc_sat(iregion,j,Cp,params);
            CT = C(j,ipg)/(C(j,iT)*R);
            CTp = Cp(j,ipg)/(Cp(j,iT)*R);
            dcdt = (CT*(1-pore(1))*C(j,i+nspecies)-...
                (1-porep(1))*CTp*Cp(j,i+nspecies))/dt;
        else
            pore = calc_sat(iregion,j,C,params);
            poreR = calc_sat(iregion,j+1,C,params);
            porep = calc_sat(iregion,j,Cp,params);
            porepR = calc_sat(iregion,j+1,Cp,params);
            CT = (C(j,ipg)/C(j,iT)+C(j+1,ipg)/C(j+1,iT))/(2*R);
            CTp = (Cp(j,ipg)/Cp(j,iT)+Cp(j+1,ipg)/Cp(j+1,iT))/(2*R);
            dcdt = w*0.5*(CT*(1-pore(1))*C(j,i+nspecies)-...
                CTp*(1-porep(1))*Cp(j,i+nspecies))/dt+...
                (1-w)*0.5*(CT*(1-poreR(1))*C(j+1,i+nspecies)-...
                CTp*(1-porepR(1))*Cp(j+1,i+nspecies))/dt;
        end
        acc = eps0(iregion)*dcdt*dx/2;
    end
end
NR = flux - gen + acc;
end
```

```
function rate = react(j,C,iregion,pore)
% Function for handling homoegenous reactions

rate = [0;0;0;0;0;0];
iT = 15; ipg = 16; ipl = 19;

sat = pore(1);
MW = 18.0152;
R1 = 83.14;    % ideal gas constant (cm3 bar/mol K)
pc = C(j,ipl)-C(j,ipg); % bar
rho_w = 1.1603-5.371e-4*C(j,iT); % density of water (g/cm3)
% vapor pressure corrected for Kelvin effect
pvap0 = exp(11.6832-3816.44/(C(j,iT)-46.13));
rkelvin = exp((pc*MW)/(rho_w*R1*C(j,iT)));
pvap = pvap0*rkelvin;

% variable identifiers
    iv1 = 2; iv2 = 4; imuw = 5; iyO2 = 11; iyw = 13; iyH2 = 14;
    iT = 15; ipg = 16; ipl = 19; iplmem = 20;

    F = 96485;            % Faraday's constant (C/mol)
    R = 8.314;            % ideal gas constant (J/mol K)
    R1 = 83.14;           % ideal gas constant (cm3 bar/mol K)
    Tref = 303.15;        % reference temperature (K)
    FRT = F/(R*C(j,iT));
    kV = 1e5;             % vapor water mass transfer coefficient
    kL = 1e3;             % liquid water mass transfer coefficient
    kevap = 100;          % prerate constant for liquid to vapor (cm/s)
    a12 = 1e5;            % electrode specific interfacial area (1/cm)

    phimtH2 = 8e3; % thiele mass transfer for hydrogen (bar cm3 s/mol)
    phimtO2 = 6e3; % thiele mass transfer for oxygen (bar cm3 s/mol)

if iregion ~= 3
    if iregion == 2 || iregion == 4
        % Hydrogen Oxidation Reaction
        alphac = 1;
        alphaa = 1;
        EAHOR = 9500; % activation energy (J/mol)
        i0HOR = 1e-3*exp((EAHOR/R)*(1/Tref-1/C(j,iT)));
        etaHOR = C(j,iv1)-C(j,iv2); % overpotential
        kHOR = (i0HOR/(2*F))*(exp(alphaa*FRT*etaHOR));
        phiHOR = sqrt(phimtH2*kHOR);
        effHOR = (3/(phiHOR^2))*(phiHOR/tanh(phiHOR)-1);
        rate(1) = effHOR*(1-sat)*(a12/(2*F))*i0HOR*...
            (C(j,ipg)*C(j,iyH2)*exp(alphaa*FRT*etaHOR) -...
           exp(-alphac*FRT*etaHOR));

        % Oxygen Reduction Reaction
        EAORR = 73269; % activation energy (J/mol)
        U0 = 4.1868*(70650+8*C(j,iT)*log(C(j,iT))-92.84*C(j,iT))/(2*F);
        i0ORR = 1.1e-8*exp((EAORR/R)*(1/Tref-1/C(j,iT)));
        etaORR = etaHOR - U0; % overpotential
        kORR = (i0ORR/(4*F))*(exp(-alphac*FRT*etaORR));
        phiORR = sqrt(phimtO2*kORR);
        effORR = (3/(phiORR^2))*(phiORR/tanh(phiORR)-1);
        rate(2) = effORR*(1-sat)*(a12/(4*F))*...
```

```
                i0ORR*(C(j,ipg)*C(j,iyO2)*...
                exp(-alphac*FRT*etaORR)-exp(alphaa*FRT*etaORR));

            % water transfer to/from membrane
            V0 = MW/rho_w;     % molar volume of water (cm3/mol)
            rate(3) = kV*(C(j,imuw) - 0.1*V0*C(j,ipg) - ...
                R*C(j,iT)*log((C(j,iyw)*C(j,ipg))/pvap0));

            % water transfer from membrane to liquid
            if C(j,iplmem) >= 0
                rate(4) = kL*(C(j,iplmem)-C(j,ipl));
            end

            % Hydrogen Peroxide Formation (ORR 2e-)
            % leads to fluoride release rate
            U0_e2 = 0.695; % standard potential (V vs. SHE)
            etaH2O2 = C(j,iv1)-C(j,iv2)-U0_e2; % overpotential
            i0H2O2 = 0.007/10000; %exchange current density (A/cm2)
            rH2O2 = effORR*(1-sat)*(a12/(2*F))*i0H2O2*...
                (C(j,ipg)*C(j,iyO2)*exp(-alphac*FRT*etaH2O2));
            kFRR = 4e-1; % rate constant for FRR from H2O2 (umol/cm2)
            rate(6) = kFRR*rH2O2;
        end
        % water generation as gas
        rate(5) = -kevap*(C(j,iyw)*C(j,ipg)-pvap)/R1/C(j,iT);
    end
end
```

```
function stefmax = stefan_maxwell(mode,j,i,C,params,pore1,pore2)
% Calculates the Stefan-Maxwell equation for species fluxes at a point j
% in the gas phase at steady state for constant T and P.
iNO2 = 7; iNH2 = 10; iyO2 = 11; iyH2 = 14; iT = 15; ipg = 16;
nregion = params(1); nspecies = params(2);
R = 83.14;    % ideal gas constant (cm3 bar/mol K)
% Knudsen diffusion
%      O2        N2       H2O       H2
MW = [31.9988; 28.014; 18.0152; 2.0159];
taug = 0.5*(pore1(7)+pore2(7));
epsg = 0.5*(pore1(6)+pore2(6));
rk = 0.5*(pore1(5)+pore2(5));
diffusion = 0;
if mode == 1
    iregion = region(1,j,params);   dx = mesh(j,C,iregion,params);
    T = (C(j,iT)+C(j-1,iT))/2;      pg = (C(j,ipg)+C(j-1,ipg))/2;
    CT = (C(j,ipg)/C(j,iT)+C(j-1,ipg)/C(j-1,iT))/2/R;
    gasmass = ((C(j,iyO2:iyH2)+C(j-1,iyO2:iyH2))/2)*MW;
    drive = (C(j,i)-C(j-1,i))/dx + ((C(j,i)+C(j-1,i))/2)*...
        ((C(j,ipg)-C(j-1,ipg))/dx)*(1-MW(i-iNH2)/gasmass)/pg;
    D = diffcoeff(pg,T);
    Deff = D/taug/epsg;
    Dk = knudsen(T,MW(i-iNH2),rk);
    Dkeff = Dk/taug/epsg;
    for k = iNO2:iNO2+nspecies-1
        if k ~= i-nspecies
            diffusion = diffusion + (C(j,i)*C(j,k)+C(j-1,i)*C(j-1,k)...
                -C(j,k+nspecies)*C(j,i-nspecies)-C(j-1,k+nspecies)*...
                C(j-1,i-nspecies))/(2*CT*Deff(k-iNO2+1,i-iyO2+1));
        end
    end
    dk = -(C(j,i-nspecies))/(CT*Dkeff);
    diffusion = diffusion + dk;
elseif mode == 2
    iregion = region(2,j,params);  dx = mesh(j,C,iregion,params);
    T = (C(j,iT)+C(j+1,iT))/2;      pg = (C(j,ipg)+C(j+1,ipg))/2;
    CT = (C(j,ipg)/C(j,iT)+C(j+1,ipg)/C(j+1,iT))/2/R;
    gasmass = ((C(j,iyO2:iyH2)+C(j+1,iyO2:iyH2))/2)*MW;
    drive = (C(j+1,i)-C(j,i))/dx + ((C(j+1,i)+C(j,i))/2)*...
        ((C(j+1,ipg)-C(j,ipg))/dx)*(1-MW(i-iNH2)/gasmass)/pg;
    D = diffcoeff(pg,T);
    Deff = D/taug/epsg;
    Dk = knudsen(T,MW(i-iNH2),rk);
    Dkeff = Dk/taug/epsg;
    for k = iNO2:iNO2+nspecies-1
        if k ~= i-nspecies
            diffusion = diffusion + (C(j,i)*C(j,k)+C(j+1,i)*C(j+1,k)...
                -C(j,k+nspecies)*C(j,i-nspecies)-C(j+1,k+nspecies)*...
                C(j+1,i-nspecies))/(2*CT*Deff(k-iNO2+1,i-iyO2+1));
        end
    end
    dk = -(C(j,i-nspecies))/(CT*Dkeff);
    diffusion = diffusion + dk;
end
stefmax = drive - diffusion;
end
```

B.5 Impedance Model Code for Case Study 1

MATLAB Code for Approach 1

```
n = 4;             % number of unknowns at each mesh point
nj = 21;           % number of mesh points
C = zeros(nj,n); % change variable

% parameters
alpha = 0.5;
sigma = 7; % S/cm
kappa = 7;  % S/cm
i0 = 1e-3;  % A/cm2
a = 1e5; % 1/cm
Cdl = 1e-7; % F/cm2
params = [alpha sigma kappa i0 a Cdl];

% operating conditions
L = 0.001; % cm
T0 = 353.15; % K
Vcell = 0.2; % V
op_cond = [L T0 Vcell];

% initial guess
C(:,1) = 0:0.5/(nj-1):0.5;
C(:,2) = 0.01;
C(:,3) = 0.5:-0.5/(nj-1):0;
C(:,4) = 0;
C_ss = steady_state(C,n,nj,params,op_cond);

%% transient
frange = logspace(-3,6,91);
for ii = 1:length(frange)
    f = frange(ii); % Hz
    T = 1/f; % period (s)
    omega = 2*pi*f; deltaV = 0.001;  % V
    n_cycle = 5;  tfinal = T*n_cycle; dt = 0.0005*T;
    time(1) = 0;  k = 1;  C = C_ss; Cp = C_ss;
    Ct(:,:,1) = reshape(C_ss,[nj,n,1]);
    while time < tfinal
        k = k+1;
        time(k) = time(k-1)+dt;
        op_cond(3) = Vcell+deltaV*cos(omega*time(k));
        C = transient(C,n,nj,params,op_cond,Cp,dt);
        Ct(:,:,k) = C;
        Cp = C;
    end
    V = reshape(Ct(1,2,:),1,size(Ct,3));
    i = reshape(Ct(end,1,:),1,size(Ct,3));
    Ir(ii) = trapz(i.*cos(omega.*time))/tfinal;
    Ij(ii) = -trapz(i.*sin(omega.*time))/tfinal;
    Vr(ii) = trapz(V.*cos(omega.*time))/tfinal;
    Vj(ii) = -trapz(V.*sin(omega.*time))/tfinal;
    Z(ii) = (Vr(ii)+1j*Vj(ii))/(Ir(ii)+1j*Ij(ii));
    clear time V i C Cp Ct
end
```

```
1   function C = steady_state(C,n,nj,params,op_cond)
2       jcount = 0;           % current iteration
3       dC = 1e-9*ones(1,n); % Delta C = small variation in value of C
4
5       rtol = 1e-6;  atol = 1e-9;
6       kerr = 1;     kerrg = 1;
7
8       itmax = 10;
9
10      while (kerr == 1 || kerrg == 1) && jcount < itmax
11          jcount = jcount+1;    % update iteration
12          CC = C;               % initialize CC
13          C = autoband(n,nj,C,dC,params,op_cond,[],0);
14          kerr = 0; kerrg = 0;
15          for j = 1:nj
16              for i = 1:n
17                  if kerr == 0 && kerrg == 0
18                      if abs(C(j,i)) > rtol*abs(CC(j,i))
19                          kerr = 1;
20                      end
21                      if kerr == 1 && abs(abs(C(j,i))<atol)
22                          kerr = 0;
23                      end
24                  end
25              end
26              for i = 1:n
27                  C(j,i) = CC(j,i)+C(j,i);
28              end
29          end
30      end
31  end
```

Function `transient` is the same as function `steady_state` except for lines 1 & 13:

```
1   function C = transient(C,n,nj,params,op_cond,Cp,dt)

13          C = autoband(n,nj,C,dC,params,op_cond,Cp,dt);
```

```
function C = autoband(n,nj,C,dC,params,op_cond,Cp,dt)

J = zeros(n*nj);
b = zeros(n*nj,1);

for j = 1:nj
    A = zeros(n,n);     % matrix of dG/dC at j-1
    B = zeros(n,n);     % matrix of dG/dC at j
    D = zeros(n,n);     % matrix of dG/dC at j+1

    % initialize G (k = 1, dC = 0)
    G = eqn(j,j,1,0,C,nj,params,op_cond,Cp,dt);

    % generate A,B,D matrices
    for k = 1:n
        eq = eqn(j,j,k,dC(k),C,nj,params,op_cond,Cp,dt);
        B(:,k) = -(eq-G)./dC(k);
        if j > 1
            eq = eqn(j,j-1,k,dC(k),C,nj,params,op_cond,Cp,dt);
            A(:,k) = -(eq-G)./dC(k);
        end
        if j < nj
            eq = eqn(j,j+1,k,dC(k),C,nj,params,op_cond,Cp,dt);
            D(:,k) = -(eq-G)./dC(k);
        end
        % construct tridiagonal matrix
        for m = 1:n
            J((m-1)*nj+j,(k-1)*nj+j) = B(m,k);
            if j > 1
                J((m-1)*nj+j,(k-1)*nj+j-1) = A(m,k);
            end
            if j < nj
                J((m-1)*nj+j,(k-1)*nj+j+1) = D(m,k);
            end
        end
        % construct solution vector
        b((k-1)*nj+j) = G(k);
    end
end

Js = sparse(J);
U = Js\b;

C = reshape(U,nj,n);

end
```

```matlab
function eq = eqn(j,jp,k,dC,C,nj,params,op_cond,Cp,dt)
    C(jp,k) = C(jp,k)+dC;

    % unknowns at each point
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4;

    % physical constants
    R = 8.314;  % J/mol K
    F = 96485;  % C/mol

    % parameters
    alpha = params(1); sigma = params(2);
    kappa = params(3); i0 = params(4);
    a = params(5); Cdl = params(6);

    % operating conditions
    L = op_cond(1); T = op_cond(2); Vcell = op_cond(3);

    dx = L/(nj-1);
    FRT = F/(R*T);

    %% Equation 1: Charge Balance
    if j == 1
        eq(ii1) = C(j,ii1);
    else
        eq(ii1) = (C(j,ii1)-C(j-1,ii1))/dx+(C(j,ii2)-C(j-1,ii2))/dx;
    end

    %% Equation 2: Ohm's Law
    if j == 1
        eq(iv1) = C(j,iv1) - Vcell;
    else
        eq(iv1) = C(j,ii1) + sigma*(C(j,iv1)-C(j-1,iv1))/dx;
    end

    %% Equation 3: Flux (no diffusion or convection)
    if j < nj
        eq(ii2) = C(j,ii2) + kappa*(C(j+1,iv2)-C(j,iv2))/dx;
    else
        eq(ii2) = C(j,ii2);
    end

    %% Equation 4: Polarization (kinetics)
    if j == 1
        eq(iv2) = C(j,iv2);
    else
        if dt == 0
            acc = 0;
        else
            acc = a*Cdl*((C(j,iv1)-C(j,iv2))-(Cp(j,iv1)-Cp(j,iv2)))/dt;
        end
        eq(iv2) = (C(j,ii2)-C(j-1,ii2))/dx +...
            a*i0*(alpha*FRT*(C(j,iv1)-C(j,iv2)))+acc;
    end
end
```

MATLAB Code for Approach 2

Lines 1-25 the same as for Approach 1.

```
26  frange = logspace(-3,6,91);
27  
28  for ii = 1:length(frange)
29      f = frange(ii); % frequency (Hz)
30      omega = 2*pi*f; % angular frequency
31      op_cond = [L T0 Vcell omega deltaV];
32      Ctilde = complex(C);
33      Ctilde = freq_response(Ctilde,n,nj,params,op_cond,C_ss);
34      Z(ii) = Ctilde(1,2)/Ctilde(end,1);
35  end
```

Function `freq_response` is the same as function `steady_state` except for lines 1 & 13.

```
1   function C = freq_response(C,n,nj,params,op_cond,C_ss)

13          C = autoband_Z(n,nj,C,dC,params,op_cond,C_ss);
```

Function `autoband_Z` is the same as function `autoband` except for lines 12,16,19,23.

```
12      G = eqn_Z(j,j,1,0,C,nj,params,op_cond,C_ss);

16          eq = eqn_Z(j,j,k,dC(k),C,nj,params,op_cond,C_ss);

19              eq = eqn_Z(j,j-1,k,dC(k),C,nj,params,op_cond,C_ss);

23              eq = eqn_Z(j,j+1,k,dC(k),C,nj,params,op_cond,C_ss);
```

```matlab
function eq = eqn_Z(j,jp,k,dC,C,nj,params,op_cond,C_ss)

    C(jp,k) = C(jp,k)+dC;

    % unknowns at each point
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4;

    % physical constants
    R = 8.314;  % J/mol K  F = 96485;  % C/mol

    % parameters
    alpha = params(1); sigma = params(2); kappa = params(3);
    i0 = params(4); a = params(5); Cdl = params(6);

    % operating conditions
    L = op_cond(1); T = op_cond(2); Vcell = op_cond(3);
    omega = op_cond(4); deltaV = op_cond(5);

    dx = L/(nj-1);
    FRT = F/(R*T);

    %% Equation 1: Charge Balance
    if j == 1
        eq(ii1) = C(j,ii1);
    else
        eq(ii1) = (C(j,ii1)-C(j-1,ii1))/dx+(C(j,ii2)-C(j-1,ii2))/dx;
    end

    %% Equation 2: Ohm's Law
    if j == 1
        eq(iv1) = C(j,iv1) - deltaV;
    else
        eq(iv1) = C(j,ii1) + sigma*(C(j,iv1)-C(j-1,iv1))/dx;
    end

    %% Equation 3: Flux (no diffusion or convection)
    if j < nj
        eq(ii2) = C(j,ii2) + kappa*(C(j+1,iv2)-C(j,iv2))/dx;
    else
        eq(ii2) = C(j,ii2);
    end

    %% Equation 4: Polarization (kinetics)
    if j == 1
        eq(iv2) = C(j,iv2);
    else
        eq(iv2) = (C(j,ii2)-C(j-1,ii2))/dx +...
            a*i0*(alpha*FRT*(C(j,iv1)-C(j,iv2)))+...
            a*Cdl*1i*omega*(C(j,iv1)-C(j,iv2));
    end
end
```

MATLAB Code for Approach 3

Lines 1-25 the same as for Approach 1 and Approach 2.

```
26  frange = logspace(-3,6,91);
27
28  for ii = 1:length(frange)
29      f = frange(ii);       % frequency (Hz)
30      omega = 2*pi*f; % angular frequency
31      op_cond = [L T0 Vcell omega deltaV];
32      Ctilde = [C_ss zeros(nj,n)];
33      Ctilde = freq_response_ReIm(Ctilde,n,nj,params,op_cond,C_ss);
34      CRe = Ctilde(:,1:n);
35      CIm = Ctilde(:,n+1:2*n);
36
37      VRe = CRe(1,2);
38      VIm = CIm(1,2);
39      iRe = CRe(end,1);
40      iIm = CIm(end,1);
41      Z(ii) = (VRe+1i*VIm)/(iRe+1i*iIm);
42  end
```

Function `freq_response_ReIm` is the same as function `steady_state` except for lines 1 & 13.

```
1   function C = freq_response_ReIm(C,n,nj,params,op_cond,C_ss)

13          C = autoband_ReIm(n,nj,C,dC,params,op_cond,C_ss);
```

Function `autoband_ReIm` is the same as function `autoband` except for lines 3-4,7-9,12,16,19,23,44.

```
3       J = zeros(2*n*nj,2*n*nj); % block tridiagonal matrix
4       b = zeros(2*n*nj,1);

7       A = zeros(2*n,2*n);      % matrix of dG/dC at j-1
8       B = zeros(2*n,2*n);      % matrix of dG/dC at j
9       D = zeros(2*n,2*n);      % matrix of dG/dC at j+1

12      G = eqn_ReIm(j,j,1,0,C,nj,params,op_cond,C_ss);

16          eq = eqn_ReIm(j,j,k,dC(k),C,nj,params,op_cond,C_ss);

19              eq = eqn_ReIm(j,j-1,k,dC(k),C,nj,params,op_cond,C_ss);

23              eq = eqn_ReIm(j,j+1,k,dC(k),C,nj,params,op_cond,C_ss);

44  C = reshape(U,nj,2*n);
```

```
function eq = eqn_ReIm(j,jp,k,dC,C,nj,params,op_cond,C_ss)
    C(jp,k) = C(jp,k)+dC;

    % unknowns at each point
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4;
    n = 4;

    % physical constants
    R = 8.314;  % J/mol K    F = 96485;  % C/mol

    % parameters
    alpha = params(1); sigma = params(2);
    kappa = params(3); i0 = params(4);
    a = params(5);    Cdl = params(6);

    % operating conditions
    L = op_cond(1);  T = op_cond(2);
    Vcell = op_cond(3); omega = op_cond(4); deltaV = op_cond(5);

    dx = L/(nj-1);
    FRT = F/(R*T);

    %% Equation 1: Charge Balance
    % Real
    if j == 1
        eq(ii1) = C(j,ii1);
    else
        eq(ii1) = (C(j,ii1)-C(j-1,ii1))/dx+(C(j,ii2)-C(j-1,ii2))/dx;
    end
    % Imaginary
    if j == 1
        eq(ii1+n) = C(j,ii1+n);
    else
        eq(ii1+n) = (C(j,ii1+n)-C(j-1,ii1+n))/dx+...
            (C(j,ii2+n)-C(j-1,ii2+n))/dx;
    end

    %% Equation 2: Ohm's Law
    % Real
    if j == 1
        eq(iv1) = C(j,iv1) - deltaV;
    else
        eq(iv1) = C(j,ii1) + sigma*(C(j,iv1)-C(j-1,iv1))/dx;
    end
    % Imaginary
    if j == 1
        eq(iv1+n) = C(j,iv1+n);
    else
        eq(iv1+n) = C(j,ii1+n) + sigma*(C(j,iv1+n)-C(j-1,iv1+n))/dx;
    end

    %% Equation 3: Flux (no diffusion or convection)
    % Real
    if j < nj
        eq(ii2) = C(j,ii2) + kappa*(C(j+1,iv2)-C(j,iv2))/dx;
    else
        eq(ii2) = C(j,ii2);
    end
    % Imaginary
    if j < nj
        eq(ii2+n) = C(j,ii2+n) + kappa*(C(j+1,iv2+n)-C(j,iv2+n))/dx;
    else
        eq(ii2+n) = C(j,ii2+n);
    end

    %% Equation 4: Polarization (kinetics)
    % Real
    if j == 1
        eq(iv2) = C(j,iv2);
    else
        eq(iv2) = (C(j,ii2)-C(j-1,ii2))/dx +...
            a*i0*(alpha*FRT*(C(j,iv1)-C(j,iv2)))-...
            a*Cdl*omega*(C(j,iv1+n)-C(j,iv2+n));
    end
    % Imaginary
    if j == 1
        eq(iv2+n) = C(j,iv2+n);
    else
        eq(iv2+n) = (C(j,ii2+n)-C(j-1,ii2+n))/dx +...
            a*i0*(alpha*FRT*(C(j,iv1+n)-C(j,iv2+n)))+...
            a*Cdl*omega*(C(j,iv1)-C(j,iv2));
    end
end
```

B.6 Impedance Model Code for Case Study 2

<u>MATLAB Code for Calculating Steady State</u>

Function `steady_state` is the same.

Function `transient` is the same.

Function `autoband` is the same.

```
function [eq,C] = eqn(j,jp,k,dC,C,nj,params,op_cond,Cp,dt)
    C(jp,k) = C(jp,k)+dC;
    % unknowns at each point
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;
    % parameters
    sigma = params(4); kappa = params(5); D = params(8);
    % operating conditions
    L = op_cond(1); Vcell = op_cond(3); C0 = op_cond(4); dx = L/(nj-1);

    %% Equation 1: Charge Balance
    if j == 1
        eq(ii1) = C(j,ii1);
    else
        eq(ii1) = (C(j,ii1)-C(j-1,ii1))/dx+(C(j,ii2)-C(j-1,ii2))/dx;
    end

    %% Equation 2: Ohm's Law
    if j == nj
        eq(iv1) = C(j,iv1) - Vcell;
    else
        eq(iv1) = C(j,ii1) + sigma*(C(j+1,iv1)-C(j,iv1))/dx;
    end

    %% Equation 2: Flux (no diffusion or convection)
    if j < nj
        eq(ii2) = C(j,ii2) + kappa*(C(j+1,iv2)-C(j,iv2))/dx;
    else
        eq(ii2) = C(j,ii2);
    end

    %% Equation 4: Polarization (kinetics)
    if j == 1
        eq(iv2) = C(j,iv2);
    else
        eq(iv2) = fluxleft(j,ii2,C,Cp,params,op_cond,dt,dx)-...
            fluxright(j,ii2,C,Cp,params,op_cond,dt,dx);
    end

    % Equation 5: Concentration Gradient (Fick's Law)
    if j == 1
        eq(iyO2) = fluxright(j,iNO2,C,Cp,params,op_cond,dt,dx);
    else
        R = 83.14; % cm3 bar / mol K
        T = op_cond(2); p = op_cond(5); CT = p/(T*R);
        eq(iyO2) = C(j,iNO2) + D*CT*(C(j,iyO2)-C(j-1,iyO2))/dx;
    end

    % Equation 6: Flux (conservation of mass)
    if j < nj
        eq(iNO2) = fluxleft(j,iNO2,C,Cp,params,op_cond,dt,dx)-...
            fluxright(j,iNO2,C,Cp,params,op_cond,dt,dx);
    else
        eq(iNO2) = C(j,iyO2) - C0;
    end
end
```

```
function NL = fluxleft(j,i,C,Cp,params,op_cond,dt,dx)
% Calculates the flux exiting the box to the left of point j
% Variable Identifiers
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;
    n = params(1);

% Flux in the box to the left
if i == ii2
    flux = C(j-1,i);
elseif i == iNO2
    flux = C(j,i);
end

% Reaction terms
F = 96845;
st = zeros(n,1); st(ii2) = -4*F; st(iNO2) = -1;
rate = react(j,C,params,op_cond);
if j ~= 1
    rateL = react(j-1,C,params,op_cond);
else
    rateL = rate;
end
w = 0.5;
gen  = st(i)*(w*rate+(1-w)*rateL)*dx/2;

if dt == 0
    acc = 0;
else
    if i == ii2
        a = params(6); Cdl = params(7);
        if j == 1
            dVdt = a*Cdl*((C(j,iv1)-C(j,iv2))-...
                 (Cp(j,iv1)-Cp(j,iv2)))/dt;
        else
            dVdt = 0.5*a*Cdl*((C(j,iv1)-C(j,iv2))-...
                 (Cp(j,iv1)-Cp(j,iv2)))/dt+...
                 0.5*a*Cdl*((C(j-1,iv1)-C(j-1,iv2))-...
                 (Cp(j-1,iv1)-Cp(j-1,iv2)))/dt;
        end
        acc = dVdt*dx/2;
    elseif i == iNO2
        R = 83.14; % cm3 bar / mol K
        T = op_cond(2); p = op_cond(5); CT = p/(T*R);
        if j == 1
            dcdt = CT*(C(j,iyO2)-Cp(j,iyO2))/dt;
        else
            dcdt = 0.5*CT*((C(j,iyO2)-Cp(j,iyO2))/dt)+...
                 0.5*CT*((C(j-1,iyO2)-Cp(j-1,iyO2))/dt);
        end
        acc = dcdt*dx/2;
    end
end

NL = flux + gen - acc;
end
```

```
function NR = fluxright(j,i,C,Cp,params,op_cond,dt,dx)
% Calculates the flux exiting the box to the right of point j
% Variable Identifiers
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;
    n = params(1); nj = params(2);

% Flux in the box to the right
if i == ii2
    flux = C(j,i);
elseif i == iNO2
    flux = C(j+1,i);
end

% Reaction terms
F = 96845; % C/mol
st = zeros(n,1); st(ii2) = -4*F; st(iNO2) = -1;
rate = react(j,C,params,op_cond);
if j ~= nj
    rateR = react(j+1,C,params,op_cond);
else
    rateR = rate;
end
w = 0.5;
gen  = st(i)*(w*rate+(1-w)*rateR)*dx/2;

if dt == 0
    acc = 0;
else
    if i == ii2
        a = params(6); Cdl = params(7);
        if j == nj
            dVdt = a*Cdl*((C(j,iv1)-C(j,iv2))-...
                (Cp(j,iv1)-Cp(j,iv2)))/dt;
        else
            dVdt = 0.5*a*Cdl*((C(j,iv1)-C(j,iv2))-...
                (Cp(j,iv1)-Cp(j,iv2)))/dt+...
                0.5*a*Cdl*((C(j+1,iv1)-C(j+1,iv2))-...
                (Cp(j+1,iv1)-Cp(j+1,iv2)))/dt;
        end
        acc = dVdt*dx/2;
    elseif i == iNO2
        R = 83.14; % cm3 bar / mol K
        T = op_cond(2); p = op_cond(5); CT = p/(T*R);
        if j == nj
            dcdt = CT*(C(j,iyO2)-Cp(j,iyO2))/dt;
        else
            dcdt = 0.5*CT*((C(j,iyO2)-Cp(j,iyO2))/dt)+...
                0.5*CT*((C(j+1,iyO2)-Cp(j+1,iyO2))/dt);
        end
        acc = dcdt*dx/2;
    end
end

NR = flux - gen + acc;
end
```

```
function rate = react(j,C,params,op_cond)
% Function for handling homoegenous reactions
ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;

T = op_cond(2); p = op_cond(5); Tref = 303.15;

% physical constants
R = 8.314;  % J/mol K
F = 96485;  % C/mol
FRT = F/(R*T);

alpha = params(3); a12 = params(6);
% exchange current density (A/cm2)
i0ORR = 1e-7*exp((73269/R)*(1/Tref-1/T));
U0 = 4.1868*(70650+8*T*log(T)-92.84*T)/(2*F); % standard potential (V)
rate = (a12/(4*F))*i0ORR*p*C(j,iyO2)*...
    exp(-alpha*FRT*(C(j,iv1)-C(j,iv2)-U0));

end
```

MATLAB Code for Approach 2

```
n = 6;                  % number of unknowns at each mesh point
nj = 21;                % number of mesh points
C = zeros(nj,n);        % change variable

% parameters
alpha = 1;
sigma = 7;    % S/cm
kappa = 0.1;  % S/cm
a = 1e3;      % 1/cm
Cdl = 2e-5;   % F/cm2
D = 0.3;      % cm2/s (O2 in water)
params = [n nj alpha sigma kappa a Cdl D];

% operating conditions
L = 0.001;    % cm
T0 = 353.15;  % K
Vcell = 1; % V
deltaV = 1e-5;
RH = 0.5;
p = 1;
Pwsat = exp(11.6832-3816.44/(T0-46.13));
C0 = 0.21*(1-RH*(Pwsat/p));
op_cond = [L T0 Vcell C0 p];

load C_ss.mat C_ss
C_ss = steady_state(C_ss,n,nj,params,op_cond);

frange = logspace(-3,4,71);
for ii = 1:length(frange)
    f = frange(ii);        % frequency (Hz)
    omega = 2*pi*f; % angular frequency
    op_cond = [L T0 deltaV C0 p omega];
    Ctilde = complex(C_ss);
    Ctilde = freq_response(Ctilde,n,nj,params,op_cond,C_ss);
    Z(ii) = Ctilde(end,2)/(Ctilde(end,1));
end
```

Function `freq_response` is the same.

Function `autoband_Z` is the same.

```matlab
function eq = eqn_Z(j,jp,k,dC,C,nj,params,op_cond,C_ss)
    C(jp,k) = C(jp,k)+dC;
    % unknowns at each point
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;
    % parameters
    sigma = params(4); kappa = params(5); D = params(8);
    % operating conditions
    L = op_cond(1); deltaV = op_cond(3); dx = L/(nj-1);

    %% Equation 1: Charge Balance
    if j == 1
        eq(ii1) = C(j,ii1);
    else
        eq(ii1) = (C(j,ii1)-C(j-1,ii1))/dx+(C(j,ii2)-C(j-1,ii2))/dx;
    end

    %% Equation 2: Ohm's Law
    if j == nj
        eq(iv1) = C(j,iv1) - deltaV;
    else
        eq(iv1) = C(j,ii1) + sigma*(C(j+1,iv1)-C(j,iv1))/dx;
    end

    %% Equation 3: Flux (no diffusion or convection)
    if j < nj
        eq(ii2) = C(j,ii2) + kappa*(C(j+1,iv2)-C(j,iv2))/dx;
    else
        eq(ii2) = C(j,ii2);
    end

    %% Equation 4: Polarization (kinetics)
    if j == 1
        eq(iv2) = C(j,iv2);
    else
        eq(iv2) = fluxleft_Z(j,ii2,C,C_ss,params,op_cond,dx)-...
            fluxright_Z(j,ii2,C,C_ss,params,op_cond,dx);
    end

    % Equation 5: Concentration Gradient (Fick's Law)
    if j == 1
        eq(iyO2) = fluxright_Z(j,iNO2,C,C_ss,params,op_cond,dx);
    else
        R = 83.14; % cm3 bar / mol K
        T = op_cond(2); p = op_cond(5); CT = p/(T*R);
        eq(iyO2) = C(j,iNO2) + D*CT*(C(j,iyO2)-C(j-1,iyO2))/dx;
    end

    % Equation 6: Flux (conservation of mass)
    if j < nj
        eq(iNO2) = fluxleft_Z(j,iNO2,C,C_ss,params,op_cond,dx)-...
            fluxright_Z(j,iNO2,C,C_ss,params,op_cond,dx);
    else
        eq(iNO2) = C(j,iyO2);
    end
end
```

```
function NL = fluxleft_Z(j,i,C,C_ss,params,op_cond,dx)
% Calculates the flux exiting the box to the left of point j
% Variable Identifiers
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;
    n = params(1);

% Flux in the box to the left
if i == ii2
    flux = C(j-1,i);
elseif i == iNO2
    flux = C(j,i);
end

% Reaction terms
F = 96845;
st = zeros(n,1);
st(ii2) = -4*F; st(iNO2) = -1;
rate = react_Z(j,C,params,op_cond,C_ss);
if j ~= 1
    rateL = react_Z(j-1,C,params,op_cond,C_ss);
else
    rateL = rate;
end
w = 0.5;
gen  = st(i)*(w*rate+(1-w)*rateL)*dx/2;

a = params(6); Cdl = params(7); omega = op_cond(6);
if i == ii2
    if j == 1
        dVdt = a*Cdl*1i*omega*(C(j,iv1)-C(j,iv2));
    else
        dVdt = 0.5*a*Cdl*1i*omega*(C(j,iv1)-C(j,iv2))+...
            0.5*a*Cdl*1i*omega*(C(j-1,iv1)-C(j-1,iv2));
    end
    acc = dVdt*dx/2;
elseif i == iNO2
    R = 83.14; % cm3 bar / mol K
    T = op_cond(2); p = op_cond(5);
    CT = p/(T*R);
    if j == 1
        dcdt = 1i*omega*CT*C(j,iyO2);
    else
        dcdt = 0.5*1i*omega*CT*C(j,iyO2)+0.5*1i*omega*CT*C(j-1,iyO2);
    end
    acc = dcdt*dx/2;
end

NL = flux + gen - acc;
end
```

```matlab
function NR = fluxright_Z(j,i,C,C_ss,params,op_cond,dx)
% Calculates the flux exiting the box to the right of point j
% Variable Identifiers
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;
    n = params(1); nj = params(2);

% Flux in the box to the right
if i == ii2
    flux = C(j,ii2);
elseif i == iNO2
    flux = C(j+1,i);
end

% Reaction terms
F = 96845; % C/mol
st = zeros(n,1);
st(ii2) = -4*F; st(iNO2) = -1;
rate = react_Z(j,C,params,op_cond,C_ss);
if j ~= nj
    rateR = react_Z(j+1,C,params,op_cond,C_ss);
else
    rateR = rate;
end
w = 0.5;
gen  = st(i)*(w*rate+(1-w)*rateR)*dx/2;

a = params(6); Cdl = params(7); omega = op_cond(6);
if i == ii2
    if j == nj
        dVdt = a*Cdl*1i*omega*(C(j,iv1)-C(j,iv2));
    else
        dVdt = 0.5*a*Cdl*1i*omega*(C(j,iv1)-C(j,iv2))+...
            0.5*a*Cdl*1i*omega*(C(j+1,iv1)-C(j+1,iv2));
    end
    acc = dVdt*dx/2;
elseif i == iNO2
    R = 83.14; % cm3 bar / mol K
    T = op_cond(2); p = op_cond(5);
    CT = p/(T*R);
    if j == nj
        dcdt = 1i*omega*CT*C(j,iyO2);
    else
        dcdt = 0.5*1i*omega*CT*C(j,iyO2)+0.5*1i*omega*CT*C(j+1,iyO2);
    end
    acc = dcdt*dx/2;
end

NR = flux - gen + acc;
end
```

```
function rate = react_Z(j,C,params,op_cond,C_ss)
% Function for handling homoegenous reactions
ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;

T = op_cond(2); p = op_cond(5); Tref = 303.15;
% physical constants
R = 8.314;  % J/mol K
F = 96485;  % C/mol
FRT = F/(R*T);

alpha = params(3); a12 = params(6);
% exchange current density (A/cm2)
i0ORR = 1e-7*exp((73269/R)*(1/Tref-1/T));
U0 = 4.1868*(70650+8*T*log(T)-92.84*T)/(2*F); % standard potential (V)
const = (a12/(4*F))*i0ORR*p;
rate = -const*C_ss(j,iyO2)*alpha*FRT*...
    exp(-alpha*FRT*(C_ss(j,iv1)-C_ss(j,iv2)-U0))*...
    (C(j,iv1)-C(j,iv2))+const*C(j,iyO2)*...
    exp(-alpha*FRT*(C_ss(j,iv1)-C_ss(j,iv2)-U0));
end
```

MATLAB Code for Approach 3

Lines 1-28 the same as for Approach 2.

```
21  for ii = 1:length(frange)
22      f = frange(ii); % frequency (Hz)
23      omega = 2*pi*f; % angular frequency
24      op_cond = [L T0 deltaV C0 p omega];
25      Ctilde = freq_response_ReIm([C_ss zeros(nj,n)],...
26          n,nj,params,op_cond,C_ss);
27      CRe = Ctilde(:,1:n);
28      CIm = Ctilde(:,n+1:2*n);
29      VRe = CRe(end,2);      VIm = CIm(end,2);
30      iRe = CRe(end,1);      iIm = CIm(end,1);
31      Z(ii)  =  (VRe+1i*VIm)/(iRe+1i*iIm);
32  end
```

Function `freq_response_ReIm` is the same.

Function `autoband_ReIm` is the same.

```
function eq = eqn_ReIm(j,jp,k,dC,C,nj,params,op_cond,C_ss)
    C(jp,k) = C(jp,k)+dC;
    % unknowns at each point
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;
    n = params(1);

    % parameters
    sigma = params(4); kappa = params(5); D = params(8);

    % operating conditions
    L = op_cond(1);   deltaV = op_cond(3);  dx = L/(nj-1);

    %% Equation 1: Charge Balance
    % Real
    if j == 1
        eq(ii1) = C(j,ii1);
    else
        eq(ii1) = (C(j,ii1)-C(j-1,ii1))/dx+(C(j,ii2)-C(j-1,ii2))/dx;
    end
    % Imaginary
    if j == 1
        eq(ii1+n) = C(j,ii1+n);
    else
        eq(ii1+n) = (C(j,ii1+n)-C(j-1,ii1+n))/dx+...
            (C(j,ii2+n)-C(j-1,ii2+n))/dx;
    end

    %% Equation 2: Ohm's Law
    % Real
    if j == nj
        eq(iv1) = C(j,iv1) - deltaV;
    else
        eq(iv1) = C(j,ii1) + sigma*(C(j+1,iv1)-C(j,iv1))/dx;
    end
    % Imaginary
    if j == nj
        eq(iv1+n) = C(j,iv1+n);
    else
        eq(iv1+n) = C(j,ii1+n) +...
            sigma*(C(j+1,iv1+n)-C(j,iv1+n))/dx;
    end

    %% Equation 3: Flux (no diffusion or convection)
    % Real
    if j < nj
        eq(ii2) = C(j,ii2) + kappa*(C(j+1,iv2)-C(j,iv2))/dx;
    else
        eq(ii2) = C(j,ii2);
    end
    % Imaginary
    if j < nj
        eq(ii2+n) = C(j,ii2+n) +...
            kappa*(C(j+1,iv2+n)-C(j,iv2+n))/dx;
    else
        eq(ii2+n) = C(j,ii2+n);
    end

    %% Equation 4: Polarization (kinetics)
    % Real
    if j == 1
        eq(iv2) = C(j,iv2);
    else
        eq(iv2) = fluxleft_ReIm(j,ii2,C,C_ss,params,op_cond,dx)-...
            fluxright_ReIm(j,ii2,C,C_ss,params,op_cond,dx);
    end
    % Imaginary
    if j == 1
        eq(iv2+n) = C(j,iv2+n);
    else
        eq(iv2+n) = fluxleft_ReIm(j,ii2+n,C,C_ss,params,op_cond,dx)-...
            fluxright_ReIm(j,ii2+n,C,C_ss,params,op_cond,dx);
    end

    %% Equation 5: Concentration Gradient (Fick's Law)
    % Real
    if j == 1
        eq(iyO2) = fluxright_ReIm(j,iNO2,C,C_ss,params,op_cond,dx);
    else
        R = 83.14; % cm3 bar / mol K
        T = op_cond(2); p = op_cond(5);
        CT = p/(T*R);
        eq(iyO2) = C(j,iNO2) + D*CT*(C(j,iyO2)-C(j-1,iyO2))/dx;
    end
    % Imaginary
    if j == 1
        eq(iyO2+n) = fluxright_ReIm(j,iNO2+n,C,C_ss,params,op_cond,dx);
    else
        R = 83.14; % cm3 bar / mol K
        T = op_cond(2); p = op_cond(5);
        CT = p/(T*R);
        eq(iyO2+n) = C(j,iNO2+n) + D*CT*(C(j,iyO2+n)-C(j-1,iyO2+n))/dx;
    end
    %% Equation 6: Flux (conservation of mass)
    % Real
    if j < nj
        eq(iNO2) = fluxleft_ReIm(j,iNO2,C,C_ss,params,op_cond,dx)-...
            fluxright_ReIm(j,iNO2,C,C_ss,params,op_cond,dx);
    else
        eq(iNO2) = C(j,iyO2);
    end
    % Imaginary
    if j < nj
        eq(iNO2+n)= fluxleft_ReIm(j,iNO2+n,C,C_ss,params,op_cond,dx)-...
            fluxright_ReIm(j,iNO2+n,C,C_ss,params,op_cond,dx);
    else
        eq(iNO2+n) = C(j,iyO2+n);
    end
end
```

```
function NL = fluxleft_ReIm(j,i,C,C_ss,params,op_cond,dx)
% Calculates the flux exiting the box to the left of point j
% Variable Identifiers
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;
    n = params(1);

% Flux in the box to the left
if i == ii2 || i == ii2+n
    flux = C(j-1,i);
elseif i == iNO2 || i == iNO2+n
    flux = C(j,i);
end

% Reaction terms
F = 96845;
st = zeros(2*n,2);
st(ii2,:) = [-4*F 0]; st(iNO2,:) = [-1 0];
st(ii2+n,:) = [0 -4*F]; st(iNO2+n,:) = [0 -1];
rate = react_ReIm(j,C,params,op_cond,C_ss);
if j ~= 1
    rateL = react_ReIm(j-1,C,params,op_cond,C_ss);
else
    rateL = rate;
end
w = 0.5;
gen  = sum(st(i,:).*(w*rate+(1-w)*rateL)*dx/2);

a = params(6); Cdl = params(7); omega = op_cond(6);
if i == ii2
    if j == 1
        dVdt = -a*Cdl*omega*(C(j,iv1+n)-C(j,iv2+n));
    else
        dVdt = -0.5*a*Cdl*omega*(C(j,iv1+n)-C(j,iv2+n))-...
            0.5*a*Cdl*omega*(C(j-1,iv1+n)-C(j-1,iv2+n));
    end
    acc = dVdt*dx/2;
elseif i == ii2+n
    if j == 1
        dVdt = a*Cdl*omega*(C(j,iv1)-C(j,iv2));
    else
        dVdt = 0.5*a*Cdl*omega*(C(j,iv1)-C(j,iv2))+...
            0.5*a*Cdl*omega*(C(j-1,iv1)-C(j-1,iv2));
    end
    acc = dVdt*dx/2;
elseif i == iNO2
    R = 83.14; % cm3 bar / mol K
    T = op_cond(2); p = op_cond(5); CT = p/(T*R);
    if j == 1
       dcdt = -omega*CT*C(j,iyO2+n);
    else
        dcdt = -0.5*omega*CT*C(j,iyO2+n)-0.5*omega*CT*C(j-1,iyO2+n);
    end
    acc = dcdt*dx/2;
elseif i == iNO2+n
    R = 83.14; % cm3 bar / mol K
    T = op_cond(2); p = op_cond(5); CT = p/(T*R);
    if j == 1
```

```
        dcdt = omega*CT*C(j,iyO2);
    else
        dcdt = 0.5*omega*CT*C(j,iyO2)+0.5*omega*CT*C(j-1,iyO2);
    end
   acc = dcdt*dx/2;
end

NL = flux + gen - acc;
end
```

```matlab
function NR = fluxright_ReIm(j,i,C,C_ss,params,op_cond,dx)
% Calculates the flux exiting the box to the right of point j
% Variable Identifiers
    ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;
    n = params(1); nj = params(2);

% Flux in the box to the right
if i == ii2 || i == ii2+n
    flux = C(j,i);
elseif i == iNO2 || i == iNO2+n
    flux = C(j+1,i);
end

% Reaction terms
F = 96845; % C/mol
st = zeros(2*n,2);
st(ii2,:) = [-4*F 0]; st(iNO2,:) = [-1 0];
st(ii2+n,:) = [0 -4*F]; st(iNO2+n,:) = [0 -1];
rate = react_ReIm(j,C,params,op_cond,C_ss);
if j ~= nj
    rateR = react_ReIm(j+1,C,params,op_cond,C_ss);
else
    rateR = rate;
end
w = 0.5;
gen  = sum(st(i,:).*(w*rate+(1-w)*rateR)*dx/2);

a = params(6); Cdl = params(7); omega = op_cond(6);
if i == ii2
    if j == nj
        dVdt = -a*Cdl*omega*(C(j,iv1+n)-C(j,iv2+n));
    else
        dVdt = -0.5*a*Cdl*omega*(C(j,iv1+n)-C(j,iv2+n))-...
            0.5*a*Cdl*omega*(C(j+1,iv1+n)-C(j+1,iv2+n));
    end
    acc = dVdt*dx/2;
elseif i == ii2+n
    if j == nj
        dVdt = a*Cdl*omega*(C(j,iv1)-C(j,iv2));
    else
        dVdt = 0.5*a*Cdl*omega*(C(j,iv1)-C(j,iv2))+...
            0.5*a*Cdl*omega*(C(j+1,iv1)-C(j+1,iv2));
    end
    acc = dVdt*dx/2;
elseif i == iNO2
    R = 83.14; % cm3 bar / mol K
    T = op_cond(2); p = op_cond(5); CT = p/(T*R);
    if j == nj
        dcdt = -omega*CT*C(j,iyO2+n);
    else
        dcdt = -0.5*omega*CT*C(j,iyO2+n)-0.5*omega*CT*C(j+1,iyO2+n);
    end
    acc = dcdt*dx/2;
elseif i == iNO2+n
    R = 83.14; % cm3 bar / mol K
    T = op_cond(2); p = op_cond(5); CT = p/(T*R);
    if j == nj
```

```
        dcdt = omega*CT*C(j,iyO2);
    else
        dcdt = 0.5*omega*CT*C(j,iyO2)+0.5*omega*CT*C(j+1,iyO2);
    end
    acc = dcdt*dx/2;
end

NR = flux - gen + acc;
end
```

```
function rate = react_ReIm(j,C,params,op_cond,C_ss)
% Function for handling homoegenous reactions
ii1 = 1; iv1 = 2; ii2 = 3; iv2 = 4; iyO2 = 5; iNO2 = 6;

T = op_cond(2); p = op_cond(5); Tref = 303.15;

% physical constants
R = 8.314;  % J/mol K
F = 96485;  % C/mol
FRT = F/(R*T);

n = params(1); alpha = params(3); a12 = params(6);
% exchange current density (A/cm2)
i0ORR = 1e-7*exp((73269/R)*(1/Tref-1/T));
U0 = 4.1868*(70650+8*T*log(T)-92.84*T)/(2*F); % standard potential (V)
const = (a12/(4*F))*i0ORR*p;
% real
rate(1) = -const*C_ss(j,iyO2)*alpha*FRT*...
    exp(-alpha*FRT*(C_ss(j,iv1)-C_ss(j,iv2)-U0))*...
    (C(j,iv1)-C(j,iv2))+const*C(j,iyO2)*...
    exp(-alpha*FRT*(C_ss(j,iv1)-C_ss(j,iv2)-U0));
% imaginary
rate(2) = -const*C_ss(j,iyO2)*alpha*FRT*...
    exp(-alpha*FRT*(C_ss(j,iv1)-C_ss(j,iv2)-U0))*...
    (C(j,iv1+n)-C(j,iv2+n))+const*C(j,iyO2+n)*...
    exp(-alpha*FRT*(C_ss(j,iv1)-C_ss(j,iv2)-U0));
end
```