Commun. Comput. Phys.
doi: 10.4208/cicp.180412.301012a

Vol. **14**, No. 3, pp. 537-573
September 2013

Using an ILU/Deflation Preconditioner for Simulation of a PEM Fuel Cell Cathode Catalyst Layer

Kyle J. Lange[1,3,*], Pang-Chieh Sui[1] and Ned Djilali[1,2]

[1] *Institute of Integrated Energy Systems, University of Victoria, BC, Canada.*
[2] *Department of Mechanical Engineering, University of Victoria, BC, Canada.*
[3] *Currently at Lawrence Livermore National Laboratory, Livermore, CA, USA.*

Received 18 April 2012; Accepted (in revised version) 30 October 2012

Available online 25 January 2013

Abstract. Numerical aspects of a pore scale model are investigated for the simulation of catalyst layers of polymer electrolyte membrane fuel cells. Coupled heat, mass and charged species transport together with reaction kinetics are taken into account using parallelized finite volume simulations for a range of nanostructured, computationally reconstructed catalyst layer samples. The effectiveness of implementing deflation as a second stage preconditioner generally improves convergence and results in better convergence behavior than more sophisticated first stage pre-conditioners. This behavior is attributed to the fact that the two stage preconditioner updates the preconditioning matrix at every GMRES restart, reducing the stalling effects that are commonly observed in restarted GMRES when a single stage preconditioner is used. In addition, the effectiveness of the deflation preconditioner is independent of the number of processors, whereas the localized block ILU preconditioner deteriorates in quality as the number of processors is increased. The total number of GMRES search directions required for convergence varies considerably depending on the preconditioner, but also depends on the catalyst layer microstructure, with low porosity microstructures requiring a smaller number of iterations. The improved model and numerical solution strategy should allow simulations for larger computational domains and improve the reliability of the predicted transport parameters. The preconditioning strategies presented in the paper are scalable and should prove effective for massively parallel simulations of other problems involving nonlinear equations.

AMS subject classifications: 65F08, 65F10, 65F15, 65F50

Key words: Deflation, PEM fuel cell, catalyst layer, pore scale model, porous media, preconditioner.

*Corresponding author. *Email addresses:* `lange9@llnl.gov` (K. Lange), `jsui@uvic.ca` (P.-C. Sui), `ndjilali@uvic.ca` (N. Djilali)

1 Introduction

Polymer Electrolyte Membrane (PEM) fuel cells have been the focus of intense research and development in the last decade due to their high energy conversion efficiency, low to zero emissions, and suitability for a broad range of applications from transportation to portable electronic devices. A PEM fuel cell is composed of a number of different layers as shown in Fig. 1. On each side of the fuel cell are flow channels, through which oxygen and hydrogen flow. The hydrogen and oxygen diffuse through a diffusion layer to the anode and cathode catalyst layers respectively. At the anode catalyst layer, hydrogen reacts to produce protons and electrons. The protons travel through the polymer membrane to the cathode catalyst layer, while the electrons go through an external circuit to do useful work. The electrons, protons and oxygen electrochemically react in the cathode catalyst layer to produce water.

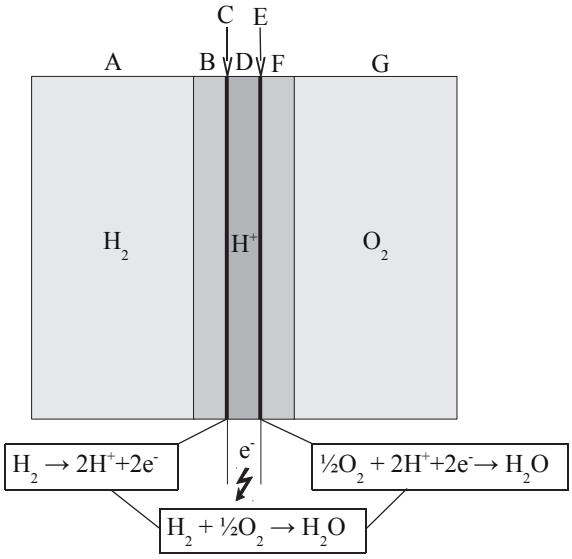


Figure 1: Simplified Schematic of PEM fuel cell operation and components: A. Anode flow channel. B. Anode gas diffusion layer. C. Anode catalyst layer. D. Polymer electrolyte membrane. E. Cathode catalyst layer. F. Cathode gas diffusion layer. G. Cathode flow channel

The operation of a PEM fuel cell relies on an array of coupled transport phenomena, including the supply of reactants, reaction kinetics, transport of ions and electrons, and removal of by-product heat and water [1]. The transport and rate limitations associated with these processes result in irreversibilities that appear as voltage losses. Many of these losses occur at the cathode catalyst layer, where the energy needed to initiate electrochemical reactions (activation polarization) is quite high [2]. A PEM fuel cell catalyst layer is a nanostructured medium composed of four distinct phases: pores which allow for reactant gas diffusion, an ionomer membrane which allows for proton conduction, carbon-black particles which allow for electron conduction, and platinum nanoparticles

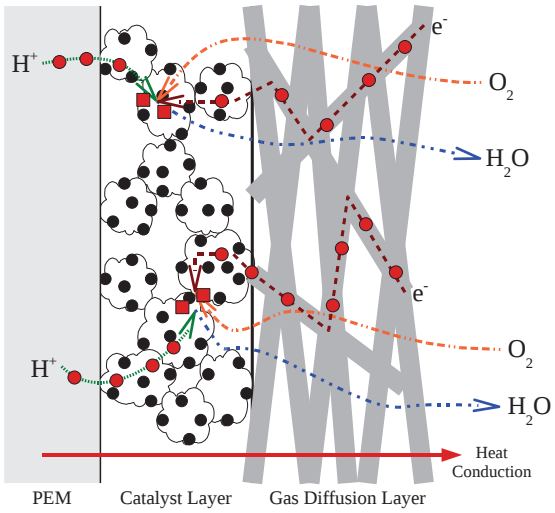


Figure 2: Schematic of species transport and heat transfer/production in the cathode catalyst layer of a PEM fuel cell. Electron transport is shown in maroon, proton transport is shown in green, oxygen diffusion is shown in orange and water vapor diffusion is shown in blue. Heat diffusion is represented by the red line. Joule heating is represented by red circles while heat generated from electrochemical reactions is shown by red squares.

which serve as catalysts for electrochemical reaction. A diagram of heat and species transport is shown in Fig. 2.

The catalyst layer is one of the most critical component of a PEM fuel cell as it strongly impacts performance, cost and durability. It is also one of the most challenging components because its complex structures, size and location make experimental observations of in-situ processes difficult to observe and measure. Computational modelling has therefore been of great interest for providing fundamental insights and guiding engineering efforts in yet characterization of the transport properties and optimization of composition and fabrication processes. Due to the small dimensions of PEM fuel cell catalyst layers (10-20 microns thick), macroscopic PEM fuel cell models do not fully resolve the microstructure of catalyst layers. Sometimes these models treat catalyst layers as infinitely thin interfaces that are described by an effective oxygen diffusivity and a bulk exchange current density [3–5]. Other models use porous electrode theory to account for the catalyst layer [2, 6–12]. More recently, macroscopic fuel cell models have assumed that catalyst layers are composed of agglomerates of carbon-black black particles which are covered by an ionomer or water layer [13–19]. Each of these approaches uses a number of simplifying assumptions to account for coupled transport phenomena and electrochemistry in catalyst layers. Some of these assumptions have been shown to be invalid (e.g. using the Bruggeman correction for the effective oxygen diffusivity in the catalyst layer [20–24]).

There is therefore a need to develop constitutive relationships for the catalyst layer which are based on fundamental physics and chemistry and which can be used in macroscopic PEM fuel cell models. These relationships must be obtained from high reso-

lution simulations which account for the complex morphology and coupled transport phenomena in this region. These include: multicomponent species diffusion, proton conduction, electron conduction, electrochemical reactions, heat transfer, ohmic heating, heat from electrochemical reactions, sorption, desorption, condensation, evaporation and two-phase flow. As might be imagined, this is quite a challenging task to account for all the relevant transport phenomena in a computational model and some assumptions are required. Most high resolution catalyst layer models that have been developed account for proton conduction, electron conduction and oxygen diffusion in a small section of the catalyst layer [21] and sometimes include electrochemical reactions [23, 25]. A recent work included water vapor transport and production along with heat transfer and heat production [26].

One of the difficulties with catalyst layer models is that there is a wide range of values for transport parameters in different phases of the catalyst layer. The thermal conductivity of air is an order of magnitude less than that of carbon black, while the diffusivity in the ionomer phase is several orders of magnitude lower than in the gas phase. Simulations involving highly contrasting values for transport coefficients, have been shown to be ill-conditioned with extremal eigenvalues that lead to poor convergence [27–30]. This issue is commonly encountered in the simulation of flow in porous media, but few papers describe preconditioning strategies that scale effectively for massively parallel simulations. One preconditioning approach used in multiphase reservoir simulations is to separate regions of the computational domain into high and low permeability regions, and solve the decoupled matrix blocks [31–34]. Domain decomposition methods have also been used for preconditioning these problems [35–37]. Another preconditioner that is often used is eigenvalue deflation [27–30, 38, 39]. Often, one of these preconditioners is used in conjunction with a standard preconditioning approach for a multi-stage preconditioner.

In a previous work [26], a numerical method was presented and used to simulate oxygen transport, water vapor transport, proton conduction, electron conduction and electrochemical reactions in a PEM fuel cell catalyst layer. While charting the path for physically based prediction of transport parameters, computational efficiency limited these simulations to small samples (200 nm x 200 nm x 200 nm), about two orders of magnitude smaller than typical catalyst layers which have a thickness of 10 microns. Larger domain simulations as well as increased resolution are for example required to account for the likely varying distributions and thickness of the ionomer. The purpose of the current work is to describe new developments to the computational model and numerical method that significantly enhance the capabilities and reliability of simulations for porous media, particularly those used in fuel cells [40]. Additional effects accounted for in the computational model are heat transfer and heat production, Stefan-Maxwell diffusion and electrochemical reactions at platinum particle sites. The numerical method is improved by using both ILU and deflation in a two-stage preconditioner, adopting a better approach to non-dimensionalizing the equations and reducing the size of the matrix.

Section 2 presents the governing equations and boundary conditions used in the model and Section 3 presents the details of the numerical method. Numerical results comparing convergence of simulations on different computational domains with different preconditioners and deflation criteria are presented in Section 4. A discussion of the results is given in Section 5 while the paper is concluded in Section 6.

2 Governing equations and boundary conditions

PEM fuel cell catalyst layers are often formed by mixing a slurry of carbon-black particles embedded with platinum with an ionomer solution and spraying the mixture onto a polymer membrane. This process results in a porous medium that is electrically and ionically conductive, but still allows for transport of reactant and product gases. A high resolution SEM image of a catalyst layer is shown in Fig. 3.

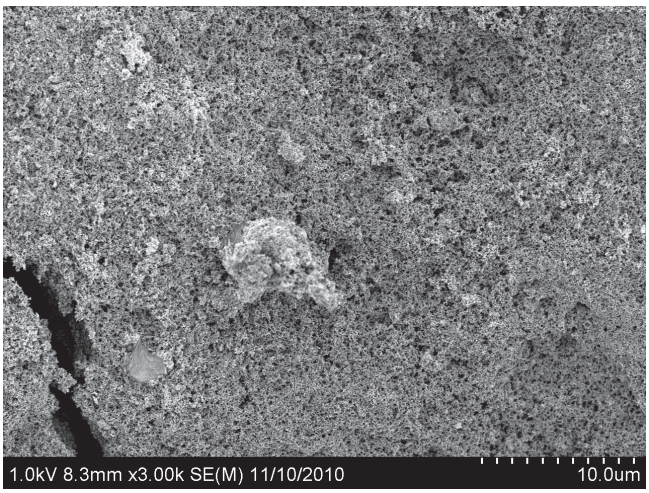


Figure 3: SEM image of the surface of a catalyst layer that has been sprayed onto a membrane.

The computational model for PEMFC catalyst layers accounts for proton conduction in the ionomer phase, electron conduction through the carbon-black particles, oxygen reduction at electrochemically active platinum particles, electro-osmotic drag of water, diffusion of oxygen and water vapor, heat conduction, ohmic heating, and heat produced by the electrochemical reactions. The model considers steady-state, isobaric, single-phase conditions in the catalyst layer. Due to small pore sizes and the fact that a gas diffusion layer is placed between the catalyst layer and the gas flow channel, convective effects are negligible. Since single phase conditions that typically prevail under low humidification conditions are considered, the evaporation, condensation, and the sorption and desorption of liquid water are not considered in the model. A uniform air pressure of 2 atmospheres representing typical operations [41–43] is assumed.

2.1 Charged particle transport and consumption

Protons conduct through the ionomer while electrons conduct through the carbon-black spheres. The conductive fluxes are computed as

$$\Gamma_{p,cond} = -\sigma_m \nabla \phi_m, \tag{2.1}$$

$$\Gamma_{e,cond} = \sigma_s \nabla \phi_s. \tag{2.2}$$

The conductivity of the carbon-black particles is taken to be 10 S/cm [44], while the conductivity of the membrane is taken from a curve-fit of experimental data for recast Nafion [45] and is computed as

$$\sigma_m \left(\mathrm{S\ cm^{-1}} \right) = c_1 \exp\left(\left[c_2 T - c_3 T^2 + c_4 T^3 - c_5 T^4 \right] a \right) + c_6, \tag{2.3}$$

where the curve-fitting parameters are given in Table 1.

Table 1: Curve-fit coefficients for membrane conductivity and saturation pressure expressions.

Coefficient	Value	Coefficient	Value
c_1	2.8133×10^{-4}	p_1	-2846.4
c_2	1.328355	p_2	411.24
c_3	-1.1642×10^{-2}	p_3	-10.554
c_4	3.442175×10^{-5}	p_4	0.16636
c_5	-3.33815×10^{-8}		
c_6	-7.2939×10^{-4}		

The relative humidity is computed as

$$a = \left(c_{H_2O} R_u T \right) / p_{sat}, \tag{2.4}$$

while the saturation pressure is computed as

$$p_{sat}\,(\mathrm{Pa}) = p_1 + p_2 (T - 273) - p_3 (T - 273)^2 + p_4 (T - 273)^3, \tag{2.5}$$

where the curve-fitting parameters are given in Table 1.

Protons and electrons are consumed in the oxygen reduction reaction and the reactive flux is expressed as

$$\Gamma_{e,r} = \Gamma_{p,r} = 1_{Pt} \left[i_0 \exp\left(\frac{-\alpha_c F}{R_u T} \eta \right) \right]. \tag{2.6}$$

The exchange current density is computed according to experimental data [46] as

$$i_0 = i_0^* \left(\frac{p_{O_2}}{p_{O_2}^*} \right)^{\gamma} \exp\left[\frac{-E_c^{rev}}{R_u T} \left(1 - \frac{T}{T^*} \right) \right]. \tag{2.7}$$

Table 2: Reaction parameters used in the model. Each parameter is taken from reference [46].

Reaction Parameter	Expression
i_0^*	2.47×10^{-8} A/cm$^2_{\text{Pt}}$
$p^*_{O_2}$	101,300 Pa
γ	0.54
E_c^{rev}	33 kJ/mol
T^*	353 K
α_c	1.0
$c^*_{O_2}$	$\frac{p^*_{O_2}}{R_u T^*}$
η	$\phi_s-\phi_m-1.3$ V

The parameters for the exchange current density and the reaction rate are given in Table 2.

Thus, the governing equations for the proton and electron potential can be expressed as the divergence of conductive and reactive fluxes:

$$\nabla\cdot\left(\Gamma_{p,cond}+\Gamma_{p,r}\right)=0, \tag{2.8}$$

$$\nabla\cdot\left(\Gamma_{e,cond}+\Gamma_{e,r}\right)=0. \tag{2.9}$$

2.2 Mass transport and production/consumption

In the pores, the Stefan-Maxwell equations are solved to compute the oxygen and water vapor diffusive fluxes. In the presence of oxygen, nitrogen, and water vapor and accounting for Knudsen diffusion, these equations take the following form in

$$\nabla y_i=\frac{R_uT}{p}\left(\sum_j^{j\neq i}\frac{y_i\Gamma_j-y_j\Gamma i}{D_{i-j}}-\frac{\Gamma_i}{D_{i.Kn}}\right), \tag{2.10}$$

where i represents a given gas species and j represents the other gas species. The binary gas diffusivities depend on the temperature and are computed from reference values taken from experiments. The governing equation for nitrogen is not solved in this model since the nitrogen mole fraction can be easily calculated by computing the nitrogen concentration and its gradient as

$$c_{N_2}=\frac{p}{R_uT}-c_{O_2}-c_{H_2O}, \tag{2.11}$$

$$\nabla c_{N_2}=-\nabla c_{O_2}-\nabla c_{H_2O}-\frac{p}{R_uT^2}\nabla T. \tag{2.12}$$

In the ionomer region, binary diffusion is assumed and thus the water vapor and oxygen diffusive fluxes in the ionomer region are computed as

$$\Gamma_{H_2O,d}=-D_{H_2O,m}\nabla c_{H_2O}, \tag{2.13}$$

$$\Gamma_{O_2,d}=-D_{O_2,m}\nabla c_{O_2}, \tag{2.14}$$

Table 3: Diffusivities used in the model.

Diffusion Coefficient	Expression
$D_{O_2-H_2O}$	$\left(\frac{0.282}{p}\right)(T/298.2)^{1.5}$ cm^2/sec [47]
$D_{O_2-N_2}$	$\left(\frac{0.220}{p}\right)(T/293.2)^{1.5}$ cm^2/sec [47]
$D_{H_2O-N_2}$	$\left(\frac{0.293}{p}\right)(T/308.1)^{1.5}$ cm^2/sec [47]
$D_{O_2,Kn}$	$(4850d)(T/32)^{0.5}$ cm^2/sec [48]
$D_{H_2O,Kn}$	$(4850d)(T/18)^{0.5}$ cm^2/sec [48]
$D_{N_2,Kn}$	$(4850d)(T/28)^{0.5}$ cm^2/sec [48]
$D_{O_2,m}$	$(0.1543(T-273)-1.65)$ cm^2/sec [49]
$D_{H_2O,m}$	$0.265a^2\exp(-3343/T)$ cm^2/sec [50]

where the diffusivities in the ionomer phase are dependent on the temperature and relative humidity. The pore and ionomer diffusivities used in this model are computed according the expressions in Table 3.

Oxygen reacts with protons and electrons at the platinum reaction sites to produce water. The oxygen and water vapor reaction fluxes are computed using Tafel kinetics as

$$\Gamma_{O_2,r}=1_{Pt}\left[\frac{1}{4F}i_0\frac{c_{O_2}}{c^*_{O_2}}\exp\left(\frac{-\alpha_c F}{R_u T}\eta\right)\right], \tag{2.15}$$

$$\Gamma_{H_2O,r}=-1_{Pt}\left[\frac{1}{2F}i_0\frac{c_{O_2}}{c^*_{O_2}}\exp\left(\frac{-\alpha_c F}{R_u T}\eta\right)\right]. \tag{2.16}$$

Water molecules are dragged by protons due to electro-osmotic drag, producing a flux which is expressed as:

$$\Gamma_{H_2O,eod}=-\frac{n_d\sigma_m\nabla\phi_m}{F}, \tag{2.17}$$

where the drag coefficient is taken to be 1 [51].

Thus, the governing equations for the concentrations of oxygen and water vapor can be expressed as the divergence of a combination of different fluxes as:

$$\nabla\cdot(\Gamma_{O_2,d}+\Gamma_{O_2,r})=0, \tag{2.18}$$

$$\nabla\cdot(\Gamma_{H_2O,d}+\Gamma_{H_2O,eod}+\Gamma_{H_2O,r})=0. \tag{2.19}$$

2.3 Heat transfer and production

Heat transfer occurs in the catalyst layer through conduction, and this flux is expressed as

$$\Gamma_{T,cond}=-k\nabla T, \tag{2.20}$$

Table 4: Temperature parameters used in the model.

Parameter	Expression
k_s W/cm-K	3.75×10^{-3} [52]
k_m W/cm-K	$\exp(0.6373a)(-.00035694(T-273)+.00165199)$ [53]
k_{air} W/cm-K	$(-.099489a+2.0)\times(.022423(T-273)+13.27)\times10^{-5}$ [54]
Π kJ/mol	$T\frac{\Delta S_h}{4F}$ [55]
S_h J/(mol-K)	326.6 [56]

where the thermal conductivities for each material are given in Table 4. The thermal conductivities for ionomer and air as a function of relative humidity and temperature are computed using curve fits of data.

Ohmic heating takes place due to proton and electron conduction and is computed as

$$S_{T,ohm}=\frac{(\nabla\phi_s)^2}{\sigma_s}+\frac{(\nabla\phi_m)^2}{\sigma_m}. \tag{2.21}$$

Finally, heat is produced via the exothermic oxygen reduction reaction and this is computed as

$$S_{T,r}=-1_{Pt}\nabla\cdot\left[i_0\frac{c_{O_2}}{c^*_{O_2}}\exp\left(\frac{-\alpha_c F}{R_u T}\eta\right)\right](\eta+\Pi), \tag{2.22}$$

where the Peltier coefficient is listed in Table 4.

Thus, the governing equation for heat transfer can be described as the divergence of the heat flux with source terms corresponding to ohmic heating and heat produced from the electrochemical reaction. This is given as

$$\nabla\cdot(\Gamma_{T,cond})=S_{T,ohm}+S_{T,r}. \tag{2.23}$$

2.4 Boundary conditions

For the two in-plane boundaries of the cubic computational domain, Dirichlet boundary conditions are used for each variable. The other four through-plane boundaries are specified to be periodic boundaries, so that species and heat transported through one boundary will come through on the opposite boundary. Because one of the primary purposes of small scale simulations such as these is to compute the effective transport properties, gradients in species concentrations, potentials and temperature are imposed in the through-plane direction across the domain. The model considers conditions of high current densities, and thus the overpotential for the electrochemical reactions are set to be approximately 0.4 Volts at each boundary. The membrane and solid potential drops across the domain are specified to make the proton and electron currents approximately equal through the domain. A temperature difference of 0.1 K is imposed, while

Table 5: Dirichlet boundary conditions for simulations.

Parameter	Boundary 1	Boundary 2
c_{O_2} (mol/m^3)	10.1	10.0
c_{H_2O} (mol/m^3)	15.9	16.0
ϕ_m (V)	1.7	1.708
ϕ_s (V)	1.3	1.3018
T (K)	353.0	353.1

the oxygen and water vapor concentrations change by 0.1 mol/m^3 across the domain. The boundary conditions are listed in Table 5.

3 Numerical method

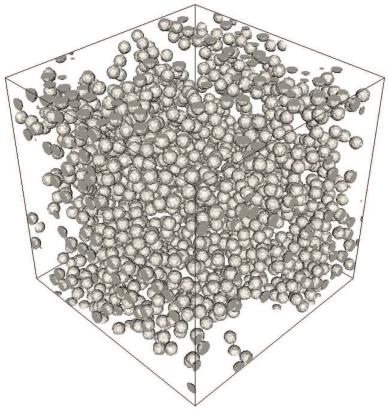
The governing equations are discretized and solved over a computational domain that is 400 nm$\times$400 nm$\times$400 nm. This represents a small piece of a PEM fuel cell catalyst layer which is normally 10-20 microns thick with an area ranging from 1-500 cm^2. In order to perform simulations, the different phases in the computational domain must be specified. It is necessary to computationally reconstruct a catalyst layer section consisting of pores, ionomer, carbon-black particles and platinum reaction sites.

3.1 Catalyst layer reconstruction

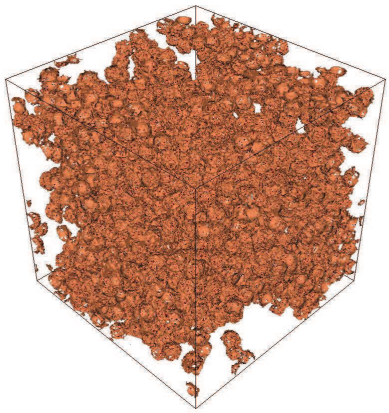
In order to numerically represent the morphology of the catalyst layer, a stochastic reconstruction algorithm based on previous works [23, 25] is used. Carbon-black spheres are randomly placed in the computational domain with a certain percentage required to be connected to other carbon-black spheres for electron conduction. Carbon-black spheres are allowed to overlap, but only by a specified tolerance. Once all of the carbon-black spheres have been placed in the domain, all cells whose cell centers fall within the radius of the carbon-black spheres are tagged as carbon-black cells. The number of ionomer cells in the domain is determined by the number of total cells in the computational domain and the specified ionomer volume fraction. The sites for the ionomer cells are stochastically chosen from available sites which are next to carbon cells and next to ionomer cells. The sites next to carbon cells are counted once, while sites next to ionomer cells can be counted multiple times to represent the increased likelihood for ionomer particles to agglomerate together. The remaining cells are tagged as pore cells.

Some examples of computationally reconstructed catalyst layer samples are shown in Figs. 4 and 5, where the computational domain is visualized by showing the carbon-black cells in grey, the ionomer cells in orange, and the pore cells in red.

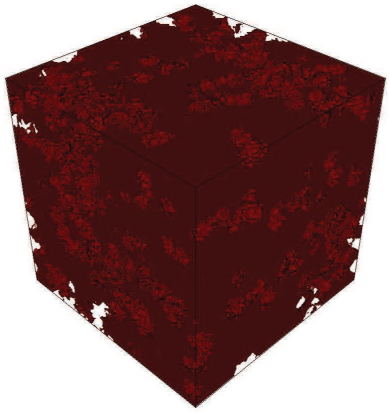
In order to allocate the platinum particles, it is assumed that they are embedded on the surface of the carbon-black particles. Because exchange current densities are given



(a) Carbon black morphology for HPM 2

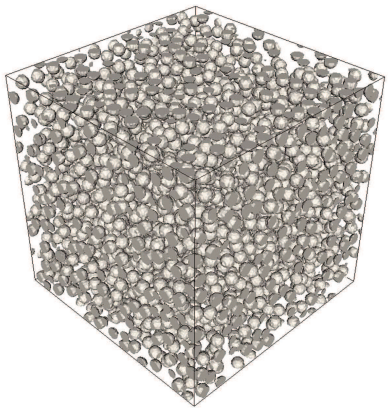


(b) Ionomer morphology for HPM 2

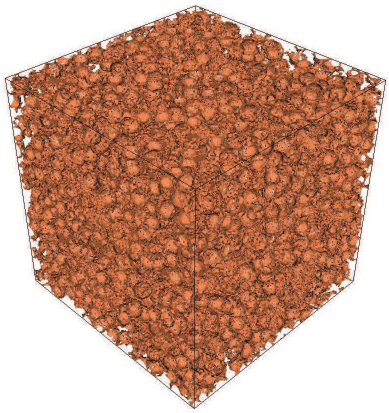


(c) Pore morphology for HPM 2

Figure 4: Morphology of HPM2.



(a) Carbon black morphology for LPM 4



(b) Ionomer morphology for LPM 4

(c) Pore morphology for LPM 4

Figure 5: Morphology of LPM4.

as current per unit area, the platinum particles are assumed to exist at the interface of a carbon cell and a non-carbon cell. The platinum loading μ_{Pt} is provided as an input to the reconstruction algorithm. Thus, the total platinum mass in the domain can be computed as:

$$m_{Pt} = \frac{\mu_{Pt} l_{dom}^3}{l_{cath}}, \tag{3.1}$$

where m_{Pt} is the total platinum mass in the domain, μ_{Pt} is the platinum loading, l is the size of the computational domain, and l_{cath} is the thickness of the catalyst layer.

Each platinum particle is assumed to be spherical. Thus, the mass of each individual platinum particle is computed as

$$m_{Pt,part} = \frac{4}{3}\pi (r_{Pt})^3 \rho_{Pt}, \tag{3.2}$$

where $m_{Pt,part}$ is the mass of each platinum particle, r_{Pt} is the platinum particle radius, and ρ_{Pt} is the density of platinum. The total number of platinum particles is computed as

$$n_{Pt} = \frac{m_{Pt}}{m_{Pt,part}}, \tag{3.3}$$

where n_{Pt} is the total number of platinum particles. Because the exchange current density is given as current per unit area, the platinum particles are assumed to exist as faces on the exterior of the carbon-black spheres. The platinum particles are randomly placed at these locations.

After the cells have been tagged as carbon-black, ionomer, or pores and the platinum particles have been placed, the mesh is analyzed to determine which cells are active, and which cells are dead. The carbon-black cells are analyzed to determine which cells are connected parts of a continuous percolating path across the computational domain. These cells are classified as active cells, which are capable of transporting electron currents and participating in electrochemical reactions. The remaining carbon-black cells are classified as dead cells, which do not transport current or participate in electrochemical reactions. A similar procedure is used to classify the ionomer cells as dead or active cells. Platinum particles which exist at the interface of an active carbon-black cell and an active ionomer cell are deemed to be active reaction sites. The remaining platinum particles are deemed to be dead platinum particles and are not electrochemically active.

3.2 Non-dimensionalization of governing equations

Because the computational domain is small relative to the size of the catalyst layer, the changes between dependant variables across the solution domain are quite small, often less than one percent of the values of the solution variables. If the equations were non-dimensionalized according to the solution values, this could lead to an ill-conditioned system due to improper scaling and could also result in significant cancellation error. In order to obtain a solution that is scaled well, the physical variables in the simulation

(oxygen concentration, water vapor concentration, membrane potential, solid potential, and temperature) are all taken to be the sum of a constant baseline value and a small perturbation. The fundamental solution variables that are computed with the simulation are the small perturbations, which can be scaled much better. Thus the fundamental nondimensionalized solution variables for each cell i are defined as

$$Q_{O_2,i}=\frac{c_{O_2,i}-c_{O_2,b}}{c_{O_2,ref}},\qquad Q_{H_2O,i}=\frac{c_{H_2O,i}-c_{H_2O,b}}{c_{H_2O,ref}}, \tag{3.4}$$

$$Q_{\phi_m,i}=\frac{\phi_{m,i}-\phi_{m,b}}{\phi_{m,ref}},\qquad Q_{\phi_s,i}=\frac{\phi_{s,i}-\phi_{s,b}}{\phi_{s,ref}},\qquad Q_{T,i}=\frac{T_i-T_b}{T_{ref}}, \tag{3.5}$$

where the subscript b refers to baseline values and the subscript ref refers to non-dimensional parameters. A list of the baseline values and non-dimensional parameters are given in Tables 6 and 7.

Table 6: Baseline values used in discretizing and non-dimensionalizing equations.

Parameter	Baseline Value
$c_{O_2,b}$ (mol/m^3)	10.0
$c_{H_2O,b}$ (mol/m^3)	16.0
$\phi_{m,b}$ (V)	1.7
$\phi_{s,b}$ (V)	1.3
T_b (K)	353.0

Table 7: Reference parameters used to non-dimensionalize equations.

Parameter	Reference Value
$c_{O_2,ref}$ (mol/m^3)	0.0005
$c_{H_2O,ref}$ (mol/m^3)	0.0005
$\phi_{m,ref}$ (V)	0.00004
$\phi_{s,ref}$ (V)	0.000009
T_{ref} (K)	0.0005
l_{ref} (nm)	400
$D_{O_2,ref}$ cm^2/sec	0.220
$D_{H_2O,ref}$ cm^2/sec	0.220
$\sigma_{m,ref}$ S/cm	0.1
$\sigma_{s,ref}$ S/cm	10.0
k_{ref} (W/cm-K)	0.003

The nondimensionalized governing equations can be expressed as

$$\nabla\cdot\left(\hat{\Gamma}_{p,cond}+\hat{\Gamma}_{p,r}\right)=\frac{l_{ref}}{\sigma_{m,ref}\phi_{m,ref}}\nabla\cdot\left(\Gamma_{p,cond}+\Gamma_{p,r}\right)=0, \tag{3.6}$$

$$\nabla\cdot\left(\hat{\Gamma}_{e,cond}+\hat{\Gamma}_{e,r}\right)=\frac{l_{ref}}{\sigma_{s,ref}\phi_{s,ref}}\nabla\cdot\left(\Gamma_{e,cond}+\Gamma_{e,r}\right)=0, \tag{3.7}$$

$$\nabla\cdot\left(\hat{\Gamma}_{O_2,d}+\hat{\Gamma}_{O_2,r}\right)=\frac{l_{ref}}{D_{O_2,ref}c_{O_2,ref}}\nabla\cdot\left(\Gamma_{O_2,d}+\Gamma_{O_2,r}\right)=0, \tag{3.8}$$

$$\nabla\cdot\left(\hat{\Gamma}_{H_2O,d}+\hat{\Gamma}_{H_2O,eod}+\hat{\Gamma}_{H_2O,r}\right)=\frac{l_{ref}}{D_{H_2O,ref}c_{H_2O,ref}}\nabla\cdot\left(\Gamma_{H_2O,d}+\Gamma_{H_2O,eod}+\Gamma_{H_2O,r}\right)=0, \tag{3.9}$$

$$\nabla\cdot\left(\hat{\Gamma}_{T,cond}\right)=\frac{l_{ref}}{k_{ref}T_{ref}}\left(S_{T,ohm}+S_{T,r}\right). \tag{3.10}$$

Henceforth, the flux terms in the equations will be assumed to be in nondimensional form and the nondimensional symbol will be dropped from the equations.

3.3 Discretization

The governing equations are discretized using the finite volume method for a cell-centered structured mesh. The governing equations are integrated over each control volume for equation i as

$$\int\int\int\nabla\cdot\Gamma_i d\Omega=\int\int\int S_i d\Omega. \tag{3.11}$$

Green's theorem can be used to transform the left hand side of Eq. (3.11) into

$$\int\int\int\nabla\cdot\Gamma_i d\Omega=\int\int\Gamma_i\cdot\vec{A}d\Gamma. \tag{3.12}$$

For each cubic cell with six faces, Eq. (3.12) results in

$$\sum_{n=1}^{n=6}\Gamma_{i,n}\cdot\vec{A}_n=S_i\Omega, \tag{3.13}$$

where Ω is the cell volume, and $\vec{A}_n$ is the outward pointing normal area vector for each face of the computational cell. Note that aside from the energy equation, the right hand side of Eq. (3.13) is equal to zero.

The conductive fluxes for charged particles, diffusive fluxes for gas species and heat conduction flux are all similar in that they are the product of a transport parameter and a gradient of a solution variable. For two adjacent cells in the z-direction k and $k+1$, a standard finite-difference formula is used to compute the gradient of solution variable i at the cell face:

$$\frac{\partial Q_i}{\partial z}=\frac{Q_i^{k+1}-Q_i^k}{z^{k+1}-z^k}. \tag{3.14}$$

Similar relationships are used to compute gradients in the x- and y-directions. For transport parameters that are dependent on the temperature, the temperature at the face is computed as the average temperature of the two adjacent cells. The relative humidity is

computed by using the average water vapor concentration and average temperature at the cell face. In the case where there is an interface between two different phases (e.g. carbon-black/pore, ionomer/pore), it is assumed that the transport is limited by the cell with the smaller transport parameter. For instance, at the interface of a pore cell and an ionomer cell, the ionomer oxygen diffusivity is used, since it is much smaller than the pore oxygen diffusivity. For cases where an ionomer cell is adjacent to a carbon-black cell, the thermal conductivity of the ionomer is used at the interface since it is lower than the thermal conductivity of carbon-black. For this case, since the water vapor concentration in carbon-black cells is zero, the water vapor concentration in the ionomer cell is used to compute the thermal conductivity at the cell interface.

For interfaces between ionomer and carbon-black cells where platinum particles are located and assumed to be electrochemically active, oxygen, protons and electrons are consumed to produce water vapor. For this case, the overpotential is computed using the solid and membrane potentials in the carbon-black and ionomer cells, respectively. The oxygen concentration in the ionomer cell is used to compute the electrochemical reaction rate. The temperature is computed as the average temperature of the two-cells.

The temperature source term due to ohmic heating is discretized by assuming that the heat produced at each face is evenly distributed to the neighboring cells. Thus, for each cell k,

$$S^k_{T,ohm}\Omega^k=\sum_{n=1}^{6}\left(\frac{(\nabla\phi_{s,n})^2}{\sigma_{s,n}}+\frac{(\nabla\phi_{m,n})^2}{\sigma_{m,n}}\right)\frac{\Omega^k}{2}, \tag{3.15}$$

where the factor of $\frac{1}{2}$ indicates that the ohmic heating at each face is evenly distributed between cell k and its neighbor.

The electrochemical reaction temperature source term is converted to a surface integral using Green's theorem as

$$\begin{aligned}&\int\int\int -1_{Pt}\nabla\cdot\left[i_0\frac{c_{O_2}}{c^*_{O_2}}\exp\left(\frac{-\alpha_c F}{R_uT}\eta\right)\right](\eta+\Pi)d\Omega\\ =&-\int\int 1_{Pt}\left[i_0\frac{c_{O_2}}{c^*_{O_2}}\exp\left(\frac{-\alpha_c F}{R_uT}\eta\right)\right]\cdot\vec{A}(\eta+\Pi)d\Gamma.\end{aligned} \tag{3.16}$$

Thus,

$$S^k_{T,r}\Omega^k=-\sum_{Pt}1_{Pt}\left[i_0\frac{c_{O_2}}{c^*_{O_2}}\exp\left(\frac{-\alpha_c F}{R_uT}\eta\right)\right]\cdot\vec{A}(\eta+\Pi). \tag{3.17}$$

The residual functions for each discretized governing equation in cell k can be expressed as a sum of fluxes through the six exterior faces of the cell with a source term contribution for the heat transfer equation:

$$R^k_{\phi_m}=\sum_{n=1}^{6}\Gamma_{p,cond,n}\cdot\vec{A}_n+\sum_{Pt}\Gamma_{p,r,n}\cdot\vec{A}_n, \tag{3.18}$$

$$R^k_{\phi_s} = \sum_{n=1}^{6} \Gamma_{e,cond,n} \cdot \vec{A}_n + \sum_{Pt} \Gamma_{e,r,n} \cdot \vec{A}_n, \tag{3.19}$$

$$R^k_{O_2} = \sum_{n=1}^{6} \Gamma_{O_2,d,n} \cdot \vec{A}_n + \sum_{Pt} \Gamma_{O_2,r,n} \cdot \vec{A}_n, \tag{3.20}$$

$$R^k_{H_2O} = \left[\sum_{n=1}^{6} (\Gamma_{H_2O,d,n} + \Gamma_{H_2O,eod,n}) + \sum_{Pt} \Gamma_{H_2O,r,n} \right] \cdot \vec{A}_n, \tag{3.21}$$

$$\begin{aligned} R^k_T = &\sum_{n=1}^{6} \Gamma_{T,cond,n} \cdot \vec{A}_n - \frac{\Omega^k}{2} \sum_{n=1}^{6} \left(\frac{(\nabla \phi_{s,n})^2}{\sigma_{s,n}} + \frac{(\nabla \phi_{m,n})^2}{\sigma_{m,n}} \right) \\ &+ \sum_{Pt} \left[i_0 \frac{c_{O_2}}{c^*_{O_2}} \exp\left(\frac{-\alpha_c F}{RT} \eta \right) \right] \cdot \vec{A} (\eta + \Pi). \end{aligned} \tag{3.22}$$

3.4 Solving the linearized system of equations

The nonlinear system of discretized governing equations can be implicitly solved using Newton's Method. This entails finding a solution where the residuals for each variable in each cell are zero with successive Newton iterations by solving the linearized system of equations:

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}} \right] \Delta \mathbf{Q} = -\mathbf{R}(\mathbf{Q}). \tag{3.23}$$

The disadvantage to this approach is that the computation of $\Delta \mathbf{Q}$ is expensive. For this reason, an inexact Newton method is used [57], where an approximate solution to Eq. (3.23) is computed for each Newton iteration. This takes much less computational work, and as the numerical solution approaches the exact solution, the convergence of Newton's method is enhanced.

Because there are five governing equations for each cell, each matrix "element" $\frac{\partial \mathbf{R^i}}{\partial \mathbf{Q^j}}$ is a five by five submatrix block as

$$\frac{\partial \mathbf{R^i}}{\partial \mathbf{Q^j}} = \begin{bmatrix} \frac{\partial R^i_{O_2}}{\partial Q^j_{O_2}} & \frac{\partial R^i_{O_2}}{\partial Q^j_{H_2O}} & \frac{\partial R^i_{O_2}}{\partial Q^j_{\phi m}} & \frac{\partial R^i_{O_2}}{\partial Q^j_{\phi s}} & \frac{\partial R^i_{O_2}}{\partial Q^j_T} \\ \frac{\partial R^i_{H_2O}}{\partial Q^j_{O_2}} & \frac{\partial R^i_{H_2O}}{\partial Q^j_{H_2O}} & \frac{\partial R^i_{H_2O}}{\partial Q^j_{\phi m}} & \frac{\partial R^i_{H_2O}}{\partial Q^j_{\phi s}} & \frac{\partial R^i_{H_2O}}{\partial Q^j_T} \\ \frac{\partial R^i_{\phi m}}{\partial Q^j_{O_2}} & \frac{\partial R^i_{\phi m}}{\partial Q^j_{H_2O}} & \frac{\partial R^i_{\phi m}}{\partial Q^j_{\phi m}} & \frac{\partial R^i_{\phi m}}{\partial Q^j_{\phi s}} & \frac{\partial R^i_{\phi m}}{\partial Q^j_T} \\ \frac{\partial R^i_{\phi s}}{\partial Q^j_{O_2}} & \frac{\partial R^i_{\phi s}}{\partial Q^j_{H_2O}} & \frac{\partial R^i_{\phi s}}{\partial Q^j_{\phi m}} & \frac{\partial R^i_{\phi s}}{\partial Q^j_{\phi s}} & \frac{\partial R^i_{\phi s}}{\partial Q^j_T} \\ \frac{\partial R^i_T}{\partial Q^j_{O_2}} & \frac{\partial R^i_T}{\partial Q^j_{H_2O}} & \frac{\partial R^i_T}{\partial Q^j_{\phi m}} & \frac{\partial R^i_T}{\partial Q^j_{\phi s}} & \frac{\partial R^i_T}{\partial Q^j_T} \end{bmatrix}. \tag{3.24}$$

For certain types of cells, some of the residual equations are not solved due to the cell material. For instance, there is no proton or electron conduction in the pores. Thus,

these matrix rows are zeroed out for rows corresponding to pore cells with a one as the diagonal entry and a zero on the right hand side. Similarly, there is no proton conduction in the carbon-black phase, and no electron conduction in the ionomer phase. One can take advantage of the fact that the equations for proton and electron transport are never both solved in the same cell. Thus, the submatrix blocks can be converted to four by four blocks, with one of the rows representing either the proton residual or the electron residual, depending on the cell tag. This results in a 36 percent savings in matrix storage. The matrix entries are computed analytically by differentiating the discretized governing equations in the code. They are verified for their accuracy using finite differences.

Additional computational savings are achieved by determining a priori which ionomer and carbon-black cells are "active", in the sense that there exists percolating paths in the material from one side of the domain to the other. Inactive cells are not able to pass current through them, and thus, the off-diagonal entries in the matrix corresponding to the proton or electron potential for these cells can be zeroed out with a "1" put on the diagonal and a "0" on the right hand side.

In order to solve the linear system of equations for each Newton iteration, the preconditioned restarted Generalized Minimal Residual (GMRES) method [58] is used. The code is parallelized using the Message Passing Interface (MPI) library [59] and it is run on 64 processors.

3.5 Preconditioning approaches

In order to accelerate the convergence of GMRES for a matrix-vector problem of the form $\mathbf{Ax}=\mathbf{b}$, a preconditioner must be implemented. For this problem, a right preconditioning approach is used [60], where one first solves

$$\mathbf{AMy}=\mathbf{b}, \tag{3.25}$$

and the solution vector is obtained as

$$\mathbf{x}=\mathbf{My}. \tag{3.26}$$

In this case the matrix $\mathbf{M}$ is a matrix that converts the original matrix vector problem into a problem that is easier to solve. A number of different preconditioners were implemented and tested for their efficiency in solving the discretized system of governing equations. Block-diagonal and ILU preconditioners seek to obtain an efficient solution to the linearized system of equations by approximating the inverse of the matrix $\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]$. Deflation preconditioners seek to shift the smallest eigenvalues of the matrix away from zero so as to obtain a better-conditioned problem that is more easily solved.

3.5.1 Block-diagonal preconditioners

A block diagonal diagonal preconditioner approximates the inverse of the original matrix $\mathbf{A}$ as

$$\mathbf{M_{bd}}=[\mathrm{diag}(\mathbf{A})]^{-1}, \tag{3.27}$$

where the inverse of the diagonal blocks is used for this case. This preconditioner is simple to implement, requires very little additional storage, and can be easily applied in a parallel computing environment. However, for large matrices, this preconditioner does not provide an adequate approximation for $\mathbf{A}^{-1}$, resulting in poor convergence.

3.5.2 Localized block ILU preconditioners

A localized block ILU preconditioner is implemented in the matrix [61]. In this case, the localized block ILU preconditioner was only implemented on the matrix entries owned by a given processor. ILU preconditioners are often defined by their level of fill, and variants of this approach exist where the process of computing the full LU decomposition is implemented but only the largest elements (ILUp) or the elements whose magnitudes are below a certain tolerance (ILUT) are kept [62]. For this simulation, the block ILU(0), ILU(1) and ILUp preconditioners are implemented. The main difference between the ILUp and ILU preconditioners is that an additional computational expense is incurred in defining the sparsity pattern for the block ILUp preconditioner. Since the matrix changes at every Newton iteration, this could result in a considerable amount of computational expense in the solution process. Thus, to make the process more efficient, the sparsity pattern is computed for the first Newton iteration and is reused for subsequent Newton iterations. The algorithm for a localized block ILU algorithm can be found in the literature and is not repeated here [63]. Localized block ILU preconditioners usually outperform block-diagonal preconditioners, but require significantly more storage. In addition, as the number of processors increases, the performance of localized block-ILU preconditioners becomes worse. In the limit of a large number of processors, a localized block ILU preconditioner is equivalent to a block-diagonal preconditioner.

3.5.3 Deflation preconditioner

It is well known that the convergence of GMRES is related to the ratio of the magnitudes of the minimum and maximum eigenvalues of a matrix. A wide spread in these values can result in poor convergence while a clustered set of eigenvalues will result in better convergence. The purpose of a deflation preconditioner is to shift the smallest eigenvalues of the matrix to a value of one, and in so doing, improve the convergence of GMRES [60]. Unfortunately, the computation of exact eigenvalues and eigenvectors for large matrices on parallel processors is unfeasible. However, approximate eigenvalues and eigenvectors can be computed by solving the eigenvalue problem from the oblique projection [38]

$$\mathbf{H_m u} = \theta \mathbf{u}, \qquad \mathbf{U} = \mathbf{V_m u}, \tag{3.28}$$

where $\mathbf{H_m}$ is the upper Hessenberg matrix constructed during the Arnoldi process in GMRES, θ is an eigenvalue of $\mathbf{H_m}$ (an approximate eigenvalue of $\mathbf{A}$), $\mathbf{u}$ is the eigenvector of the upper hessenberg matrix, $\mathbf{V_m}$ is the orthogonal set of m basis vectors generated by the Arnoldi process, and $\mathbf{U}$ is the set of approximate eigenvectors of the matrix $\mathbf{A}$.

An alternative method is to use the harmonic projection to solve the following generalized eigenvalue problem [39]:

$$\mathbf{R_m u} = \theta[\mathbf{Q_m V_{m+1}^* M V_m}]\mathbf{u} \qquad \mathbf{U} = \mathbf{M V_m u}, \tag{3.29}$$

where

$$\mathbf{Q_m}\bar{\mathbf{H}} = \begin{pmatrix} \mathbf{R_m} \\ 0 \end{pmatrix}, \tag{3.30}$$

and $\mathbf{M}$ is the preconditioner corresponding to the existing set of eigenvalues as

$$\mathbf{M} = \mathbf{I_n} + \mathbf{U T^{-1} U^T}, \qquad \mathbf{AMU} = \mathbf{UT}. \tag{3.31}$$

Note that in Eq. (3.29), the square brackets indicate that the last row of the matrix product is ignored. The harmonic projection has been shown to do a better job of preconditioning than the oblique projection [64], and thus is implemented in the code using the QZ algorithm. For every iteration of restarted GMRES, n deflation vectors are constructed, up to a maximum of p deflation vectors, which are stored in the set $\mathbf{U}$. Once the maximum number of deflation vectors has been reached, the following generalized eigenvalue problem is solved to determine the eigenvectors corresponding to the minimum eigenvalues

$$(\mathbf{AU_1})^* \mathbf{AU_1 u} = \theta(\mathbf{AU_1})^* \mathbf{U_1 u}, \qquad \mathbf{U_2} = \mathbf{U_1 u}, \tag{3.32}$$

where U_1 is the basis of previously computed eigenvectors and new eigenvectors, and U_2 is the new basis of eigenvectors. More details about this approach can be found in the literature [39].

Deflation preconditioners have been shown to work well in improving the convergence of linear solvers for problems involving flow in porous media where extremal eigenvalues exist [30, 65]. Deflation preconditioners are easy to implement in a parallel environment and the amount of required memory is quite small compared to ILU preconditioners. The additional work required in implementing the preconditioner consists of a series of dot products and computation on small matrices. In many cases, they have been used as part of a two-stage preconditioner [34, 65].

3.5.4 Two-stage preconditioning implementation

A two-stage right preconditioner involves the implementation of two preconditioners M_1 and M_2 such that

$$\mathbf{AM_1 M_2 z} = \mathbf{b}. \tag{3.33}$$

The solution vector is then obtained as

$$\mathbf{y} = \mathbf{M_2 z}, \qquad \mathbf{x} = \mathbf{M_1 y}. \tag{3.34}$$

For this problem, the two stage preconditioner is constructed by using a block-diagonal or localized block ILU preconditioner as the first preconditioner and the deflation preconditioner as the second preconditioner. This makes logical sense because the preconditioners complement one another in their purpose and construction. ILU preconditioners

provide an approximate inverse to the matrix A, but require significant amounts of storage. The quality of the preconditioner deteriorates with increasing numbers of processors. On the other hand, deflation preconditioners are not designed to be an approximate inverse to the matrix A, but to shift the extremal eigenvalues towards one. They are easily parallelizable, and the performance is independent of the number of processors.

3.5.5 Restarted GMRES implementation

One of the challenges of using a deflation preconditioner is designing a strategy for obtaining optimum convergence. If the simulation is converging quickly without the deflation preconditioner, it may not be necessary to implement the deflation stage of the two-stage preconditioner. Another possibility is that a particular set of deflation vectors is effective in converging the simulation, and thus it is desirable in that case not to have any additional deflation vectors. There may be a case where deflation is not working well, and the existing deflation vectors need to be thrown out and the deflation process needs to be restarted from scratch. In addition, one must account for the fact that with Newton's method, the matrix changes throughout the solution process, and the deflation vectors computed for one matrix may not work effectively for another matrix.

Taking into account all of these possibilities, an adaptive deflation algorithm is developed to enhance the convergence of restarted GMRES. The rms ratio δ is defined for a given GMRES restart as

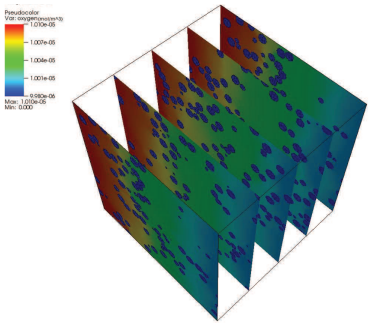
$$\delta = \frac{\|\mathbf{r}^s\|}{\|\mathbf{r}^e\|}, \tag{3.35}$$

where the subscripts s and e represent the starting and ending residual vectors. The rms ratio when deflation is not implemented is denoted as δ_{noD}. Several other parameters are defined as the absolute lower threshold $\delta_{\min}$, the relative lower threshold γ, the absolute higher threshold $\delta_{\max}$, and the cycling restart number n_c. The following algorithm is then used to obtain good convergence for restarted GMRES using a two-stage preconditioner with deflation:

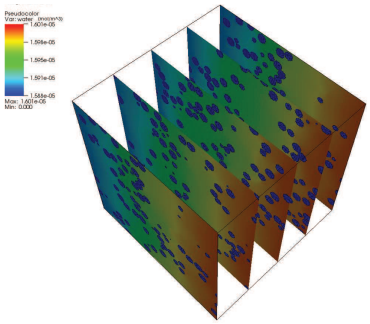
1. For the first instance of restarted GMRES, do not use deflation, but compute δ_{noD}.
2. If $\delta < \delta_{\min}$, make no changes to the second stage preconditioner.
3. If $\delta < \gamma\delta_{noD}$, make no changes to the second stage preconditioner.
4. If $\delta > \delta_{\max}$ and the maximum number of deflation vectors has been reached, throw out all existing deflation vectors and restart without deflation vectors.
5. If $\delta > \delta_{noD}$ and $m \bmod n_c = 0$, throw out all existing deflation vectors and start over.
6. If none of the above apply, add n new deflation vectors up to a maximum of p deflation vectors.

4 Results and discussion

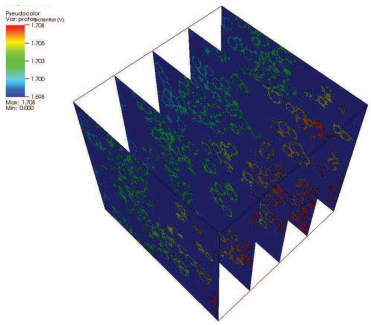
Five different low porosity microstructures (LPM) and five different high porosity microstructures (HPM) were generated for this study and a number of different parametric studies were performed for different preconditioners, GMRES search directions and deflation parameters (n, p, n_c, $\delta_{\max}$, $\delta_{\min}$, γ). The convergence behavior for each case was analyzed. In order to provide a measure for comparison, the total number of iterations and total runtime in each case is determined by the total number of search directions re-



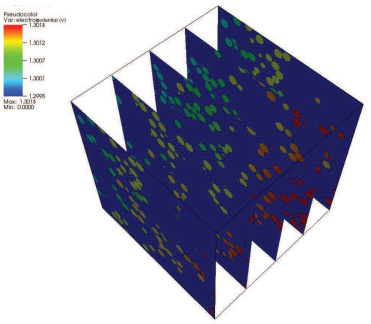
(a) Oxygen concentration for HPM 2



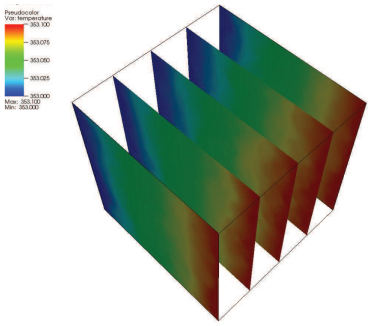
(b) Water vapor concentration for HPM 2



(c) Proton potential for HPM 2

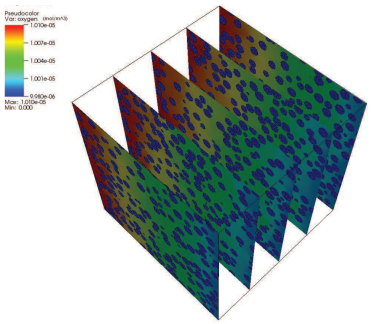


(d) Electron potential for HPM 2

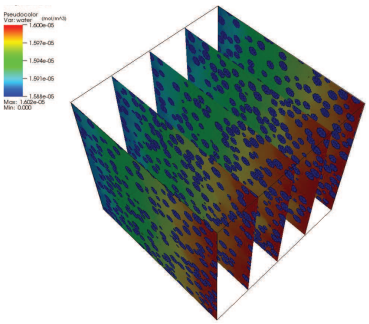


(e) Temperature for HPM 2

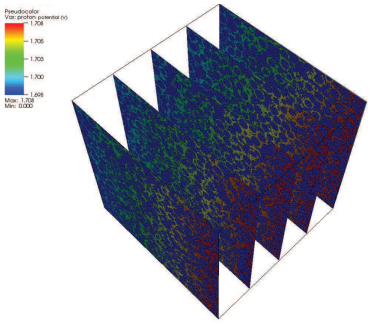
Figure 6: Solution profiles for HPM 2.



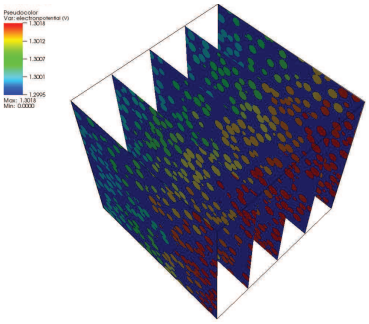
(a) Oxygen concentration for LPM 4



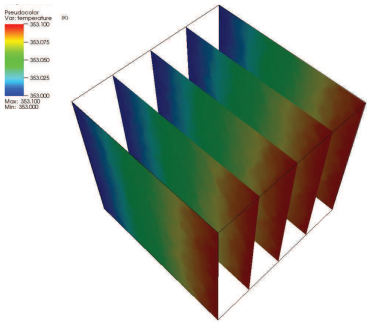
(b) Water vapor concentration for LPM 4



(c) Proton potential for LPM 4



(d) Electron potential for LPM 4



(e) Temperature for LPM 4

Figure 7: Solution profiles for LPM 4.

quired to obtain a linear residual that is less than 8×10^{-7}. For each case, the matrix had approximately 350 million nonzero entries. Solution contours for several geometries are shown in Figs. 6 and 7.

Each case was run for a maximum of 2 hours, with 3 restarted GMRES cycles for each Newton iteration, and 100 GMRES search directions for each GMRES cycle. In the cases where the number of GMRES search directions was changed, the number of Newton iterations was changed accordingly so that there was still a maximum of 12,000 total search directions. The baseline parameters are given in Table 8.

The convergence behavior for each case is compared as a function of computational

Table 8: Baseline parameters used for each simulation. Each parametric study investigates the effect of changing one of these parameters.

Parameter	Value
Preconditioner	ILUp-D
n	2
p	4
n_c	5
$\delta_{\max}$	0.9
$\delta_{\min}$	0.3
γ	0.5

runtime and the number of search directions. Due to the computational expense of running each case, the runtimes were only computed from one simulation. In order to make more definitive statements about the efficiency of each approach, multiple simulations would have to be run to compute average runtimes.

4.1 Comparison of preconditioners

Four different preconditioners were used as the first level preconditioner: block-diagonal (BD), localized block ILU(0) (LBILU(0)), localized block ILU(1) (LBILU(1)), and localized block ILUp(18) (LBILUp(18)). Cases were run with and without deflation as a second stage preconditioner (-D). The total number of iterations required for convergence in each case is shown in Tables 9 and 10, while the total amount of runtime for each case is shown in Tables 11 and 12.

Tables 9 and 10 show that the total number of GMRES search directions required for convergence varies considerably depending on the preconditioner that is used, whether

Table 9: Number of total iterations for convergence to a linear residual of 8×10^{-7} for different preconditioners on different low and high porosity (LPM and HPM) geometries. A value of N is given for the case of a preconditioner that did not converge within 2 hours of runtime.

Preconditioner	BD	BD-D	LBILU(0)	LBILU(0)-D
HPM1 Iterations	N	N	N	9986
HPM2 Iterations	N	N	N	15823
HPM3 Iterations	N	14561	14669	6767
HPM4 Iterations	N	N	N	7791
HPM5 Iterations	N	N	N	7754
LPM1 Iterations	N	17346	14273	5600
LPM2 Iterations	N	N	15311	6780
LPM3 Iterations	N	N	10292	5447
LPM4 Iterations	N	16434	10396	5149
LPM5 Iterations	N	19672	8253	4467

Table 10: Number of total iterations for convergence to a linear residual of 8×10^{-7} for different preconditioners on different geometries. A value of N is given for the case of a preconditioner that did not converge within 2 hours of runtime.

Preconditioner	LBILU(1)	LBILU(1)-D	LBILUp(18)	LBILUp(18)-D
HPM1 Iterations	11481	6059	9328	6498
HPM2 Iterations	N	9092	N	10126
HPM3 Iterations	6793	4632	6726	3691
HPM4 Iterations	9407	5758	7780	4059
HPM5 Iterations	9960	6169	6086	4872
LPM1 Iterations	6897	3289	2929	3326
LPM2 Iterations	6311	4569	8122	4062
LPM3 Iterations	5967	3477	4788	3079
LPM4 Iterations	6817	3834	6192	2556
LPM5 Iterations	6136	2989	4465	2968

or not a high porosity or low porosity microstructure is considered, and the specific details of each microstructure. Low porosity microstructures require a small number of iterations to reach convergence compared to high porosity microstructures. This has been observed previously for a very similar numerical problem [63]. The three preconditioners which obtained convergence within two hours for very case were LBILU(0)-D, LBILU(1)-D, and LBILUp(18)-D. In general, as the complexity of the preconditioner is increased, the total number of iterations required for convergence decreases as well. In most cases, the implementation of deflation as a second stage preconditioner significantly reduces the total number of search directions required for convergence.

In order to compare the computational efficiency of each algorithm, it is important to consider the runtime required for convergence. Adding deflation as a second stage preconditioner reduces the number of iterations required for convergence, but it also adds additional computational overhead. Similarly, using LBILU(1) and LBILUp(18) as a first stage preconditioner results in a smaller number of iterations required for convergence compared to LBILU(0), but the additional memory requirements and the larger stencil required for LBILU(1) and LBILUp(18) increase the computational runtime. Table 11 and Table 12 show that as far as the total amount of runtime is concerned, LBILUp(18)-D normally converges simulations in the shortest amount of time, while LBILU(0)-D and LBILU(1)-D are not far behind.

HPM2 took the longest time to converge for the high porosity microstructures, while LPM2 took the longest time to converge for the low porosity microstructure. Plots of the convergence behavior as a function of time for HPM2 and LPM2 are shown in Fig. 8 and Fig. 9 respectively. It is evident in each case that the implementation of deflation as a second stage preconditioner significantly improves the convergence with time. The time convergence of HPM1 and LMP1 are shown in Figs. 10 and 11. For HPM1, LBILU(1)-D converged the quickest, while for LPM1, LBILUp(18) converged the quickest.

Table 11: Amount of runtime required for convergence to a linear residual of 8×10^{-7} for different preconditioners on different geometries.

Preconditioner	BD	BD-D	LBILU(0)	LBILU(0)-D
HPM1 Time (sec)	N	N	N	3985
HPM2 Time (sec)	N	N	N	6519
HPM3 Time (sec)	N	5100	5534	2631
HPM4 Time (sec)	N	N	N	3176
HPM5 Time (sec)	N	N	N	3126
LPM1 Time (sec)	N	6780	5903	2181
LPM2 Time (sec)	N	N	5710	2679
LPM3 Time (sec)	N	N	3848	2125
LPM4 Time (sec)	N	5949	3586	2092
LPM5 Time (sec)	N	6718	2957	1669

Table 12: Amount of runtime required for convergence to a linear residual of 8×10^{-7} for different preconditioners on different geometries.

Preconditioner	LBILU(1)	LBILU(1)-D	LBILUp(18)	LBILUp(18)-D
HPM1 Time (sec)	6292	3218	4804	3965
HPM2 Time (sec)	N	5312	N	5931
HPM3 Time (sec)	3803	2766	3964	2392
HPM4 Time (sec)	5172	3171	4441	2473
HPM5 Time (sec)	5116	3257	3371	2927
LPM1 Time (sec)	3814	1752	1699	1993
LPM2 Time (sec)	3592	2634	4500	2479
LPM3 Time (sec)	3371	2004	2601	1882
LPM4 Time (sec)	3698	2134	3220	1742
LPM5 Time (sec)	3284	1722	2484	1780

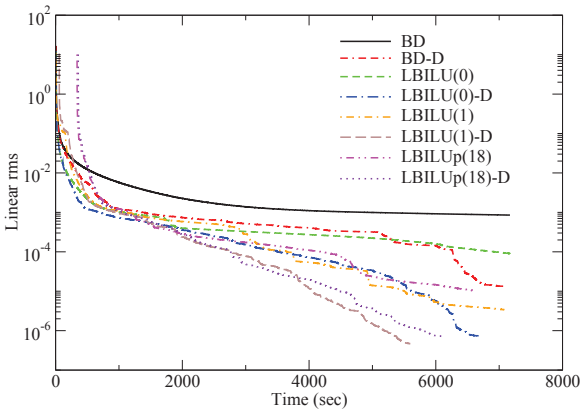


Figure 8: Plot of linear residual as a function of time for different preconditioners for HPM2.

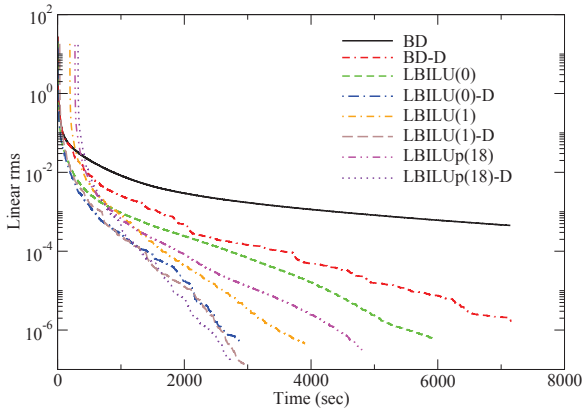


Figure 9: Plot of linear residual as a function of time for different preconditioners for LPM2.

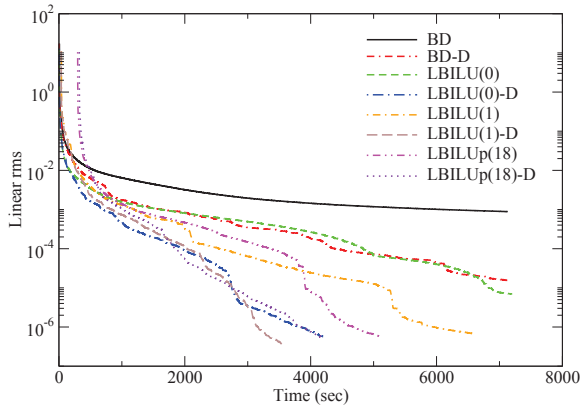


Figure 10: Plot of linear residual as a function of time for different preconditioners for HPM2.

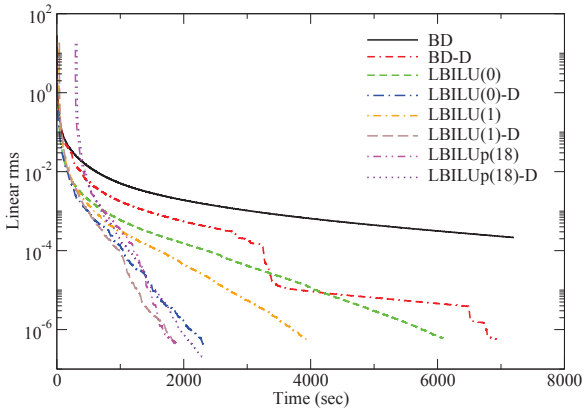


Figure 11: Plot of linear residual as a function of time for different preconditioners for LPM2.

4.2 Comparison of number of deflation vectors

The number of deflation vectors computed at each GMRES iteration (n) and the total number of deflation vectors (p) was changed for each case, and the number of iterations required for convergence are shown in Tables 13 and 14, while the total amount of time required for convergence is shown in Tables 15 and 16. There is a wide spread in the results. Clearly, the values of n and p are important parameters in determining the convergence of the simulations. As n and p increase, the total amount of computational work required to implement the deflation preconditioner increases as well, due to additional floating point operations and communication that is required.

The combination of $n=2$ and $p=4$ gave the shortest time for convergence in fifty percent of the cases. Using high values for n and p usually took 30-40 percent longer to converge compared to $n=2$ and $p=4$. However, for LPM1, having no deflation vectors resulted in the shortest convergence time. A plot of the convergence of HPM3 with time is shown in Fig. 12. Deflation improves the convergence, especially when lower values of n and p are used.

Table 13: Number of total iterations for convergence to a linear residual of 8×10^{-7} for different numbers of deflation vectors on different geometries.

Deflation Vectors	0/0	2/1	4/1	4/2
HPM1 Iterations	9328	6679	6798	6498
HPM2 Iterations	N	8755	11575	10126
HPM3 Iterations	6726	4098	3934	3691
HPM4 Iterations	7780	5079	5842	4059
HPM5 Iterations	6086	5287	6196	4872
LPM1 Iterations	2929	2851	3156	3326
LPM2 Iterations	8122	4097	4067	4062
LPM3 Iterations	4788	3496	3369	3079
LPM4 Iterations	6192	3492	3091	2556
LPM5 Iterations	4465	3280	2848	2968

Table 14: Number of total iterations for convergence to a linear residual of 8×10^{-7} for different numbers of deflation vectors on different geometries.

Deflation Vectors	8/2	8/4	16/4	16/8
HPM1 Iterations	8446	4995	8793	4726
HPM2 Iterations	9489	9047	10000	8289
HPM3 Iterations	4398	4151	4951	4582
HPM4 Iterations	5181	4899	6916	4493
HPM5 Iterations	5297	5372	6683	5878
LPM1 Iterations	3339	3379	3490	3384
LPM2 Iterations	4776	4482	5189	4722
LPM3 Iterations	3082	3094	3110	3877
LPM4 Iterations	3369	3039	3391	3459
LPM5 Iterations	3214	3189	3270	3599

Table 15: Amount of runtime (sec) required for convergence to a linear residual of 8×10^{-7} for different numbers of deflation vectors on different geometries.

Deflation Vectors	0/0	2/1	4/1	4/2
HPM1 Time (sec)	4804	3585	4051	3965
HPM2 Time (sec)	N	5027	6401	5931
HPM3 Time (sec)	3964	2634	2322	2392
HPM4 Time (sec)	4441	2865	3451	2473
HPM5 Time (sec)	3371	3097	3736	2927
LPM1 Time (sec)	1699	1761	1930	1993
LPM2 Time (sec)	4500	2656	2509	2479
LPM3 Time (sec)	2601	2084	1972	1882
LPM4 Time (sec)	3220	2036	1848	1742
LPM5 Time (sec)	2484	1991	1771	1780

Table 16: Amount of runtime (sec) required for convergence to a linear residual of 8×10^{-7} for different numbers of deflation vectors on different geometries.

Deflation Vectors	8/2	8/4	16/4	16/8
HPM1 Time (sec)	4975	3314	5331	3385
HPM2 Time (sec)	5306	5546	6205	5815
HPM3 Time (sec)	2618	2815	3289	3176
HPM4 Time (sec)	3225	3109	4115	3280
HPM5 Time (sec)	3160	3515	4376	3998
LPM1 Time (sec)	1975	2097	2320	2395
LPM2 Time (sec)	2727	2755	3117	3210
LPM3 Time (sec)	1936	2018	2175	2593
LPM4 Time (sec)	2124	1979	2219	2525
LPM5 Time (sec)	2006	2022	2241	2461

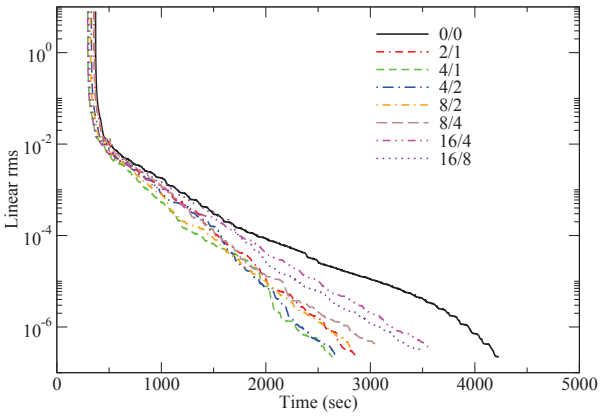


Figure 12: Plot of linear residual as a function of time for different numbers of deflation vectors for HPM3.

4.3 Comparison of restart parameters

Tables 17 and 18 show the total number of iterations required for convergence for each of the ten microstructures when the restart parameter values are changed. The baseline case refers to the simulation using the parameters listed in Table 8. The constant case refers to a case where deflation vectors are computed for every GMRES restart and Newton iteration and are never thrown out.

The implementation of the second stage deflation preconditioner at every GMRES restart and Newton iteration (Constant) results in poor convergence. In most cases, increasing the value of γ did not improve the convergence of the simulations. However, using a value of $\gamma=0.8$ reduced the number of required iterations by 20-25 percent for HPM1 and HPM2. Decreasing the value of $\delta_{\max}$ to 0.8 had mixed results, with only five of the ten microstructures showing a decrease in the total number of iterations. Increasing the value of $\delta_{\max}$ to 0.95 only resulted in one of the ten microstructures showing a decrease in the total number of iterations required for convergence. Changing the cycling

Table 17: Number of total iterations for convergence to a linear residual of 8×10^{-7} for different restart parameters on different geometries.

RP	Baseline	Constant	$\gamma=0.7$	$\gamma=0.8$	$\delta_{\max}=0.8$
HPM1 Iterations	6498	N	6498	5091	5294
HPM2 Iterations	10126	N	10126	7639	10126
HPM3 Iterations	3691	5378	3965	3597	3665
HPM4 Iterations	4059	N	3999	3928	4670
HPM5 Iterations	4872	N	4976	4948	4549
LPM1 Iterations	3326	3538	3326	2839	2996
LPM2 Iterations	4062	7222	4062	4451	4315
LPM3 Iterations	3079	4458	3123	3094	2962
LPM4 Iterations	2556	4890	2556	2556	2556
LPM5 Iterations	2968	4048	2968	2968	2832

Table 18: Number of total iterations for convergence to a linear residual of 8×10^{-7} for different restart parameters on different geometries.

RP	Baseline	$\delta_{\max}=0.95$	$n_c=10$	$\delta_{\min}=0.5$	$\delta_{\min}=0.7$
HPM1 Iterations	6498	7689	7337	5291	5759
HPM2 Iterations	10126	7378	11254	6967	7183
HPM3 Iterations	3691	3691	4190	3658	3777
HPM4 Iterations	4059	4882	6057	5161	4441
HPM5 Iterations	4872	5843	5383	4881	5066
LPM1 Iterations	3326	3326	3399	3519	2977
LPM2 Iterations	4062	4392	3962	3801	4252
LPM3 Iterations	3079	3079	3811	2966	3088
LPM4 Iterations	2556	2556	3470	3251	3381
LPM5 Iterations	2968	2985	2764	2983	3026

frequency to $n_c = 10$ resulted in an increase in the total number of iterations required for convergence for eight out of the ten microstructures. Increasing the absolute lower threshold $\delta_{\min}$ reduced the total number of iterations required for convergence for HPM1 and HPM2, but did not significantly reduce the number of required iterations for other microstructures.

4.4 Comparison of the number of GMRES search directions

Tables 19 and 20 show a comparison of the total number of iterations and total amount of runtime required for convergence when the total number of GMRES iterations used at each restart is changed. For every case except HPM2, the total amount of runtime does not change significantly when the total number of GMRES search directions is changed. When 50 GMRES search directions are used, the total number of iterations increases, but the amount of required storage and computational work for each GMRES cycle is significantly reduced. At higher numbers of GMRES search directions, the reverse is

Table 19: Number of iterations required for convergence to a linear residual of 8×10^{-7} for different numbers of GMRES search directions on different geometries.

GMRES search directions	50	100	150	200
HPM1 Iterations	7854	6498	5593	5299
HPM2 Iterations	14271	10126	5573	4936
HPM3 Iterations	4678	3691	3870	3109
HPM4 Iterations	5745	4059	4429	3781
HPM5 Iterations	5906	4872	4735	4167
LPM1 Iterations	4879	3326	2866	2363
LPM2 Iterations	5198	4062	3537	3206
LPM3 Iterations	3795	3079	2808	2643
LPM4 Iterations	3156	2556	2673	2329
LPM5 Iterations	3653	2968	2684	2429

Table 20: Amount of runtime required for convergence to a linear residual of 8×10^{-7} for different numbers of GMRES search directions on different geometries.

GMRES search directions	50	100	150	200
HPM1 Time (sec)	3893	3965	3722	4102
HPM2 Time (sec)	6239	5931	3724	4151
HPM3 Time (sec)	2379	2392	2797	2498
HPM4 Time (sec)	2836	2473	3083	3274
HPM5 Time (sec)	2947	2927	3244	3557
LPM1 Time (sec)	2475	1993	2127	1947
LPM2 Time (sec)	2667	2479	2479	2537
LPM3 Time (sec)	1917	1882	1992	2106
LPM4 Time (sec)	1654	1742	1878	1908
LPM5 Time (sec)	1901	1780	1902	1956

true, where more storage and computational work are required for fewer GMRES cycles. For HPM2, using 50 and 100 GMRES search directions is not sufficient to achieve good convergence.

5 Discussion

Implementing deflation as a second stage preconditioner improves the convergence for all but one of the considered cases. It also usually results in better convergence behavior than a more sophisticated first stage preconditioner. LBILU(0)-D outperforms LBILU(1) and LBILUp(18) in most cases. This is likely due to the fact that the effectiveness of the deflation preconditioner is independent of the number of processors, whereas the localized block LBILU preconditioner deteriorates in quality as the number of processors is increased. Additionally, LBILU(0)-D requires much less storage than LBILU(1) and LBILUp(18), reducing the computational overhead of implementing the preconditioner. The BD-D preconditioner did not perform as well as LBILU(0) in most cases. However, there are a number of different parameters that might be altered in the second stage deflation preconditioner, which might result in equivalent performance. Nonetheless, simulations which use matrix-free GMRES may benefit from using a two-stage block-diagonal deflation preconditioner, since only the diagonal blocks and the deflation vectors are required to be stored, and the performance is much better than a simple block-diagonal preconditioner.

It is clear from Table 17, that the implementation of the second stage deflation preconditioner at every GMRES restart and Newton iteration (Constant) results in poor convergence. In previous works [38, 39], it is shown that when not enough GMRES iterations are used at each restart, implementing deflation at each restart will not result in good convergence. The fact that only 100 search directions are used for a matrix with a dimension of 32 million indicates that this is likely what is happening here. Another interesting observation is that in previous works [38, 39], increasing the number of deflation vectors improved the convergence of restarted GMRES. However, in this case, the convergence did not improve significantly when the number of deflation vectors was increased.

This then begs the question as to why the convergence of the simulation is improved by implementing the algorithm in Section 3.5.5. In the solution process, the approximate eigenvalues obtained from solving equation 3.29 were observed to be two to three orders of magnitude less than the approximate eigenvalues obtained from equation 3.32. The improvement in convergence can be attributed to the fact that with the preconditioning matrix changing at every GMRES restart and Newton iteration, this reduces the effects of stalling often seen with restarted GMRES, where the same preconditioning matrix is used for every GMRES restart. In essence, a different linear problem with different eigenspectra is solved at every GMRES restart.

The large variance in the results in Table 15, Table 16, Table 17 and Table 18 illustrates the challenge of developing a deflation preconditioner with optimum parameters.

Making adjustments to the restart parameters and the number of deflation vectors will improve the convergence of some simulations but not others depending of the details of the reconstructed microstructure.

Future numerical work will focus on refining the adaptive deflation algorithm to obtain optimum convergence. In addition, the LBILU algorithm will be modified so that the localized block ILU decomposition will use matrix entries from neighboring processors as well. On the physical modelling front, future developments will need to account for two-phase flow to extend the applicability to a broader range of operating conditions. Finally, multi-scaling strategies are required to couple pore scale models to macroscopic models of complete fuel cells which have spatial resolution that are several orders of magnitude coarser.

6 Conclusions

The preconditioning strategies presented are scalable and should prove effective for massively parallel simulations of other non-linear porous media problems. For most of the considered PEMFC catalyst layer simulations, implementing deflation as a second-stage preconditioner with dynamic restarting improves the convergence of restarted GMRES. The convergence behavior will vary depending on the number of deflation vectors and the values of the restart parameters. Changing the number of GMRES search directions did not significantly affect the total runtime for most simulations. It is unlikely that the improved convergence is not due to an accurate approximation of an eigenvector of the matrix, but rather due to the changing of the matrix problem at each GMRES restart.

Acknowledgments

This work was funded through the Natural Science and Engineering Research Council (NSERC) Discovery Grant program and the Canada Research Chairs Program. Computational resources were provided by WestGrid, a member of Compute Canada.

Nomenclature

Variables

1_{Pt}	value is 1 at platinum reaction sites and 0 elsewhere
$\vec{A}$	outward normal pointing area vector
$\mathbf{A}$	matrix for linear problem
a	relative humidity
$\mathbf{b}$	right hand side of linear problem
c	gas concentration
c_1	membrane conductivity curve-fitting parameter
c_2	membrane conductivity curve-fitting parameter

c_3	membrane conductivity curve-fitting parameter
c_4	membrane conductivity curve-fitting parameter
c_5	membrane conductivity curve-fitting parameter
c_6	membrane conductivity curve-fitting parameter
D	gas diffusivity
E_c^{rev}	activation energy
F	Faraday's constant
H_m	Hessenberg matrix constructed during Arnoldi process of GMRES
$\mathbf{I_m}$	Identity matrix
i_0	exchange current density
k	thermal conductivity
l	length
$\mathbf{M}$	preconditioning matrix for linear problem
m	mass
n	number
n_c	cycling restart number
n_d	electro-osmotic drag coefficient
p	pressure
p_1	saturation pressure curve-fitting parameter
p_2	saturation pressure curve-fitting parameter
p_3	saturation pressure curve-fitting parameter
p_4	saturation pressure curve-fitting parameter
$\mathbf{Q}$	solution variable vector
$\mathbf{Q_m}$	orthonormal matrix formed from Q-R decomposition of $\mathbf{H_m}$
$\mathbf{R}$	residual vector
$\mathbf{R_m}$	upper triangular matrix formed from Q-R decomposition of $\mathbf{H_m}$
R_u	Universal gas constant
r	radius
S	source term
$\mathbf{T}$	matrix used in deflation
T	Temperature
$\mathbf{U}$	approximate eigenvector of $\mathbf{A}$
$\mathbf{u}$	deflation eigenvector
$\mathbf{V}$	orthogonal set of basis vectors generated by the Arnoldi Process
$\mathbf{x}$	solution vector for linear problem
y	mole fraction
$\mathbf{y}$	intermediate vector for preconditioned linear problem
z	z position
$\mathbf{z}$	intermediate vector two-stage preconditioned linear problem

Greek Symbols

α_c	cathodic charge transfer coefficient
Γ	flux
γ	reaction order
δ	rms ratio
η	overpotential
θ	deflation eigenvalue
μ	platinum loading

Π	Peltier heat coefficient
ρ	density
σ	conductivity
ϕ	potential
Ω	volume

Subscripts

b	baseline
bd	block diagonal
$cath$	cathode
$cond$	conductive
d	diffusive
dom	computational domain
e	electron
eod	electro-osmotic drag
H_2O	water vapor
i	solution variable index
Kn	Knudsen
ILU	incomplete LU
m	membrane
max	maximum
min	minimum
n	face index
N_2	nitrogen
noD	no Deflation
O_2	oxygen
$part$	particle
p	proton
Pt	platinum
$reac$	reactive
ref	reference
s	solid
sat	saturation
T	heat

Superscripts

$*$	reference value for electrochemical reaction
e	end
i	cell index
j	cell index
k	cell index
s	start

References

[1] N. Djilali and P. Sui, Transport phenomena in fuel cells: From microscale to macroscale, Int. J. Comput. Fluid Dyn., 22 (2008), 115–133.

[2] D. H. Schwarz and N. Djilali, 3d modeling of catalyst layers in pem fuel cells, J. Electrochem. Soc., 154 (2007), B1167–B1178.
[3] T. E. Springer, T. A. Zawodzinski and S. Gottesfeld, Polymer electrolyte fuel cell model, J. Electrochem. Soc., 138 (1991), 2334–2342.
[4] T. V. Nguyen and R. E. White, A water and heat management model for proton-exchange-membrane fuel cells, J. Electrochem. Soc., 140 (1993), 2178–2186.
[5] J. H. Nam and M. Kaviany, Effective diffusivity and water-saturation distribution in single- and two-layer pemfc diffusion medium, Int. J. Heat Mass Trans., 46 (2003), 4595–4611.
[6] D. Bernardi and M. Verbrugge, Mathematical model of a gas diffusion electrode bonded to a polymer electrolyte, AICHE J., 37 (1991), 1151–1163.
[7] D. M. Bernardi and M. W. Verbrugge, A mathematical model of the solid-polymer-electrolyte fuel cell, J. Electrochem. Soc., 139 (1992), 2477–2491.
[8] T. F. Fuller and J. Newman, Water and thermal management in solid-polymer-electrolyte fuel cells, J. Electrochem. Soc., 140 (1993), 1218–1225.
[9] A. A. Kulikovsky, J. Divisek and A. A. Kornyshev, Modeling the cathode compartment of polymer electrolyte fuel cells: Dead and active reaction zones, J. Electrochem. Soc., 146 (1999), 3981–3991.
[10] R. Bradean, K. Promislow and B. Wetton, Transport phenomena in the porous cathode of a proton exchange membrane fuel cell, Numer. Heat Transfer, Part A, 42 (2002), 121–138.
[11] J. Yuan, M. Rokni and B. Sundén, A numerical investigation of gas flow and heat transfer in proton exchange membrane fuel cells, Numer. Heat Transfer, Part A, 44 (2003), 255–280.
[12] T. Berning and N. Djilali, A 3D, multiphase, multicomponent model of the cathode and anode of a pem fuel cell, J. Electrochem. Soc., 150 (2003), A1589–A1598.
[13] R. P. Iczkowski and M. B. Cutlip, Voltage losses in fuel cell cathodes, J. Electrochem. Soc., 127 (1980), 1433–1440.
[14] Y. W. Rho, S. Srinivasan and Y. T. Kho, Mass transport phenomena in proton exchange membrane fuel cells using o2/he, o2/ar, and o2/n2 mixtures, J. Electrochem. Soc., 141 (1994), 2089–2096.
[15] O. Antoine, Y. Bultel, R. Durand and P. Ozil, Electrocatalysis, diffusion and ohmic drop in pemfc: Particle size and spatial discrete distribution effects, Electrochim. Acta, 43 (1998), 3681–3691.
[16] H. P. L. H. van Bussel, F. G. H. Koene and R. K. A. M. Mallant, Dynamic model of solid polymer fuel cell water management, J. Power Sources, 71 (1998), 218–222.
[17] L. Pisani, G. Murgia, M. Valentini and B. D'Aguanno, A working model of polymer electrolyte fuel cells, J. Electrochem. Soc., 149 (2002), A898–A904.
[18] M. Secanell, K. Karan, A. Suleman and N. Djilali, Multi-variable optimization of pemfc cathodes using an agglomerate model, Electrochim. Acta, 52 (2007), 6318–6337.
[19] P. K. Das, X. Li and Z.-S. Liu, A three-dimensional agglomerate model for the cathode catalyst layer of pem fuel cells, J. Power Sources, 179 (2008), 186–199.
[20] G. Wang, P. P. Mukherjee and C.-Y. Wang, Direct numerical simulation (dns) modeling of pefc electrodes: Part ii. random microstructure, Electrochim. Acta, 51 (2006), 3151–3160.
[21] S. H. Kim and H. Pitsch, Reconstruction and effective transport properties of the catalyst layer in pem fuel cells, J. Electrochem. Soc., 156 (2009), B673–B681.
[22] Z. Yu and R. N. Carter, Measurement of effective oxygen diffusivity in electrodes for proton exchange membrane fuel cells, J. Power Sources, 195 (2010), 1079–1084.
[23] K. J. Lange, P.-C. Sui and N. Djilali, Pore scale simulation of transport and electrochemical reactions in reconstructed pemfc catalyst layers, J. Electrochem. Soc., 157 (2010), B1434–B1442.

[24] J. Shen, J. Zhou, N. G. Astrath, T. Navessin, Z.-S. S. Liu, C. Lei, J. H. Rohling, D. Bessarabov, S. Knights and S. Ye, Measurement of effective gas diffusion coefficients of catalyst layers of pem fuel cells with a loschmidt diffusion cell, J. Power Sources, 196 (2011), 674–678.
[25] N. Siddique and F. Liu, Process based reconstruction and simulation of a three-dimensional fuel cell catalyst layer, Electrochim. Acta, 55 (2010), 5357–5366.
[26] K. J. Lange, P.-C. Sui and N. Djilali, Pore scale modeling of a proton exchange membrane fuel cell catalyst layer: Effects of water vapor and temperature, J. Power Sources, 196 (2011), 3195–3203.
[27] C. Vuik, A. Segal and J. A. Meijerink, An efficient preconditioned cg method for the solution of a class of layered problems with extreme contrasts in the coefficients, J. Comput. Phys, 152 (1999), 385–403.
[28] J. Frank and C. Vuik, On the construction of deflation-based preconditioners, SIAM J. Sci. Comput., 23 (2001), 442–462.
[29] C. Vuik, A. Segal, L. el Yaakoubi and E. Dufour, A comparison of various deflation vectors applied to elliptic problems with discontinuous coefficients, Appl. Num. Math., 41 (2002), 219–233.
[30] R. Nabben and C. Vuik, A comparison of deflation and coarse grid correction applied to porous media flow, SIAM J. Numer. Anal., 42 (2004), 1631–1647.
[31] A. Behie and P. K. W. Vinsome, Block iterative methods for fully implicit reservoir simulation, SPE Journal, 22 (1982), 658–668.
[32] C. N. Dawson, H. Klie, M. F. Wheeler and C. S. Woodward, A parallel, implicit, cell-centered method for two-phase flow with a preconditioned Newton Krylov solver, Comp. Geosciences, 1 (1997), 215–249.
[33] R. Scheichl, R. Masson and J. Wendebourg, Decoupling and block preconditioning for sedimentary basin simulations, Comp. Geosciences, 7 (2003), 295–318.
[34] B. Aksoylu and H. Klie, A family of physics-based preconditioners for solving elliptic equations on highly heterogeneous media, Appl. Num. Math., 59 (2009), 1159–1186.
[35] J. H. Bramble, J. E. Pasciak and A. H. Schatz, An iterative method for elliptic problems on regions partitioned into substructures, Math. Comput., 46 (1986), 361–369.
[36] J. Aarnes and T. Y. Hou, Multiscale domain decomposition methods for elliptic problems with high aspect ratios, Acta Math. Appl. Sinica, 18 (2002), 63–76.
[37] I. G. Graham, P. O. Lechner and R. Scheichl, Domain decomposition for multiscale pdes, Numer. Math., 106 (2007), 589–626.
[38] J. Erhel, K. Burrage and B. Pohl, Restarted gmres preconditioned by deflation, J. Comput. Appl. Math., 69 (1996), 303–318.
[39] K. Burrage and J. Erhel, On the performance of various adaptive preconditioned gmres strategies, Numer. Linear Algebra Appl., 5 (1998), 101–121.
[40] P. Sun, G. Xue, C. Wang and J. Xu, Fast numerical simulation of two-phase transport model in the cathode of a polymer electrolyte fuel cell, Communications in Computational Physics, 6 (2008), 49–71.
[41] S. Lee, S. Mukerjee, J. McBreen, Y. Rho, Y. Kho and T. Lee, Effects of nafion impregnation on performances of pemfc electrodes, Electrochim. Acta, 43 (1998), 3693–3701.
[42] Q. Yan, H. Toghiani and J. Wu, Investigation of water transport through membrane in a pem fuel cell by water balance experiments, J. Power Sources, 158 (2006), 316–325.
[43] J. Ge, A. Higier and H. Liu, Effect of gas diffusion layer compression on pem fuel cell performance, J. Power Sources, 159 (2006), 922–927.
[44] N. Probst and E. Grivei, Structure and electrical properties of carbon black, Carbon, 40

(2002), 201–205.
[45] Y. Sone, P. Ekdunge and D. Simonsson, Proton conductivity of nafion 117 as measured by a four-electrode ac impedance method, J. Electrochem. Soc., 143 (1996), 1254–1259.
[46] K. C. Neyerlin, W. Gu, J. Jorne and H. A. Gasteiger, Determination of catalyst unique parameters for the oxygen reduction reaction in a pemfc, J. Electrochem. Soc., 153 (2006), A1955–A1963.
[47] E. Cussler, Diffusion: Mass Transfer in Fluid Systems, Cambridge University Press, 1997.
[48] E. A. Mason, A. P. Malinauskas and R. B. E. III, Flow and diffusion of gases in porous media, J. Chem. Phys., 46 (1967), 3199–3216.
[49] K. Lee, A. Ishihara, S. Mitsushima, N. Kamiya and K. ichiro Ota, Effect of recast temperature on diffusion and dissolution of oxygen and morphological properties in recast nafion, J. Electrochem. Soc., 151 (2004), A639–A645.
[50] Q. Zhao, P. Majsztrik and J. Benziger, Diffusion and interfacial transport of water in nafion, J. Phys. Chem. B, 115 (2011), 2717–2727.
[51] T. A. Zawodzinski, J. Davey, J. Valerio and S. Gottesfeld, The water content dependence of electro-osmotic drag in proton-conducting polymer electrolytes, Electrochim. Acta, 40 (1995), 297–302.
[52] J.-B. Donnet, R. C. Bansal and M.-J. Wang, Carbon Black: Science and Technology, Marcel Dekker, Inc., New York, NY, USA, 1993.
[53] M. Khandelwal and M. Mench, Direct measurement of through-plane thermal conductivity and contact resistance in fuel cell materials, J. Power Sources, 161 (2006), 1106–1115.
[54] Wolfram—Alpha, Thermal conductivity of moist air, `http://www.wolframalpha.com`, 2011.
[55] A. Z. Weber and J. Newman, Coupled thermal and water management in polymer electrolyte fuel cells, J. Electrochem. Soc., 153 (2006), A2205–A2214.
[56] M. J. Lampinen and M. Fomino, Analysis of free energy and entropy changes for half-cell reactions, J. Electrochem. Soc., 140 (1993), 3537–3546.
[57] R. S. Dembo, S. C. Eisenstat and T. Steihaug, Inexact newton methods, SIAM J. Numer. Anal., 19 (1982), 400–408.
[58] Y. Saad and M. H. Schultz, Gmres: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Stat. Comput., 7 (1986), 856–869.
[59] A. Grama, A. Gupta, G. Karypis and V. Kumar, Introduction to Parallel Computing, Addison-Wesley, 2003.
[60] K. Chen, Matrix Preconditioning Techniques and Applications, Cambridge University Press, Cambridge, UK, 2005.
[61] K. Nakajima and H. Okuda, Parallel iterative solvers with localized ilu preconditioning for unstructured grids on workstation clusters, Int. J. Comput. Fluid Dyn., 12 (1999), 315–322.
[62] Y. Saad, Iterative Methods for Sparse Linear Systems, SIAM, 2000.
[63] K. J. Lange, C. Misra, P.-C. Sui and N. Djilali, A numerical study on preconditioning and partitioning schemes for reactive transport in a pemfc catalyst layer, Comput. Meth. Appl. Mech. Eng., 200 (2011), 905–916.
[64] A. Chapman and Y. Saad, Deflated and augmented krylov subspace techniques, Numer. Linear Algebra Appl., 4 (1997), 43–66.
[65] J. Tang and C. Vuik, Efficient deflation methods applied to 3-D bubbly flow problems, Electron. Trans. Numer. Anal., 26 (2007), 330–349.