Commun. Comput. Phys.
doi: 10.4208/cicp.051010.180311a

Vol. **11**, No. 1, pp. 65-98
January 2012

Numerical Study of a 3D Two-Phase PEM Fuel Cell Model Via a Novel Automated Finite Element/Finite Volume Program Generator

Pengtao Sun[1,*], Su Zhou[2], Qiya Hu[3] and Guoping Liang[4]

[1] *Department of Mathematical Sciences, University of Nevada, Las Vegas, 4505 Maryland Parkway, Las Vegas, NV 89154, USA.*
[2] *Department of Fuel Cell Power Systems, Tongji University (Jiading campus), 4800 Caoan Road, Shanghai 201804, China.*
[3] *Institute of Computational Mathematics and Scientific Engineering Computing, Chinese Academy of Sciences, Beijing 100080, China.*
[4] *Institute of Mathematics, Academy of Mathematics and Systems Sciences, Chinese Academy of Sciences, Beijing 100080, China.*

Received 5 October 2010; Accepted (in revised version) 18 March 2011

Available online 5 September 2011

Abstract. Numerical methods of a 3D multiphysics, two-phase transport model of proton exchange membrane fuel cell (PEMFC) is studied in this paper. Due to the coexistence of multiphase regions, the standard finite element/finite volume method may fail to obtain a convergent nonlinear iteration for a two-phase transport model of PEMFC [49, 50]. By introducing Kirchhoff transformation technique and a combined finite element-upwind finite volume approach, we efficiently achieve a fast convergence and reasonable solutions for this multiphase, multiphysics PEMFC model. Numerical implementation is done by using a novel automated finite element/finite volume program generator (FEPG). By virtue of a high-level algorithm description language (script), component programming and human intelligence technologies, FEPG can quickly generate finite element/finite volume source code for PEMFC simulation. Thus, one can focus on the efficient algorithm research without being distracted by the tedious computer programming on finite element/finite volume methods. Numerical success confirms that FEPG is an efficient tool for both algorithm research and software development of a 3D, multiphysics PEMFC model with multicomponent and multiphase mechanism.

AMS subject classifications: 65B99, 65K05, 65K10, 65N08, 65N12, 65N22, 65N30, 65Z05

Key words: Proton exchange membrane fuel cell, two-phase transport, Kirchhoff transformation, finite element, finite volume, automated program generator, algorithm description language, script.

*Corresponding author. *Email addresses:* pengtao.sun@unlv.edu (P. Sun), suzhou@tongji.edu.cn (S. Zhou), hqy@lsec.cc.ac.cn (Q. Hu), guopingliang@yahoo.com.cn (G. Liang)

1 Introduction

Fuel cells have been called the key to abundant energy from secure and renewable sources, e.g., fuel cells promise to replace the internal combustion engine in transportation due to their higher energy efficiency and zero or ultralow emissions. Hydrogen proton exchange membrane fuel cell (PEMFC) is presently considered as a potential type of fuel cells for such application. Since PEMFC simultaneously involves electrochemical reactions, current distribution, two-phase flow transport and heat transfer, an extensive mathematical modeling of multiphysics system combining with the advanced numerical techniques shall make a significant impact in gaining a fundamental understanding of the interacting electrochemical and transport phenomena and providing a computer-aided tool for design and optimization of future fuel cell engines.

Modeling and numerical simulation of hydrogen PEMFC have been attempted by a number of groups with the common goal of better understanding and hence optimizing fuel cell systems. Excellent reviews of hydrogen PEMFC research up to the mid-1990s were presented in [15, 39]. Recently, a comprehensive review of fuel cell science and technology was given in [60] which summarized the fundamental models for fuel cell engineering including single-phase and multiphase models. Single-phase model is the simplest approach in which the gas and liquid are considered as a single-fluid mixture and thus share the same velocity field. This approach is suited for fuel cell simulations under low humidity operation. The more rigorous approach to liquid water transport is a true multiphase model in which the two phases travel at different velocities. However, multiphase transport in fuel cells is always a challenge in fuel cell modeling. Multiphase flow, which especially exists at high-humidity operations, originates from water production by the oxygen reduction reaction, and the produced liquid water affects gaseous reactant supply and electrochemical catalyst activity as well.

There are two types of multiphase model existing for PEMFC modeling: multifluid approach and multiphase mixture (M^2) formation [61]. In contrast to those drawbacks of multifluid model [30]: a relatively large number of primary variables for each phase, highly nonlinear equations, numerical complexity due to explicitly track the irregular and moving interface between two phases, the M^2 model is more suitable for two-phase PEMFC modeling. One major advantage of the M^2 model over the classical multifluid model is that it eliminates the need for tracking phase interfaces, thus simplifying the numerical complexity of two-phase flow and transport modeling. Moreover, the M^2 model is mathematically equivalent to multifluid models without invoking any additional approximations. Therefore, we adopt M^2 formation as the two-phase transport model of PEMFC in this paper.

Comparing to the relatively plentiful literature on modeling and experimental study of fuel cells, there are less study contributing to the numerical method of two-phase transport PEMFC model. in [67], the volume-of-fluid (VOF) method is employed for PEMFC in conjunction with an interface reconstruction algorithm to track the dynamics of the deforming water droplets. However, VOF technique particularly deals with the interface

of liquid and gas and leaves the entire flow computation to commercial computational fluid dynamics (CFD) package. Basically, the PEMFC model it deals with still belongs to multifluid model. Recently, the lattice Boltzmann method (LBM) is used to simulate the flow through an idealized PEMFC porous transport layer [11] and model the fluid flow of reactive mixtures in randomly generated porous media by simulating the actual coupling interaction among the species [6]. Whereas, LBM is based on a mesoscopic point of view, very different from the way of handling macroscopic M^2 continuum transport model. Another uncommon numerical method for PEMFC modeling is moving least squares method (MLS) studied in [56, 57]. The use of MLS is justified by the fact that it is useful for approximating experimental data. Basically, this method approaches the numerical model from the experimental phenomenology of the PEMFC. Thus, the accuracy of MLS severely relies on the sufficiency of experimental data.

Actually, most of numerical simulations done for the transport model of fuel cell in numerous literatures [22–26, 33, 34, 46, 60, 62–65] are implemented by using commercial CFD software, such as Fluent, CFD-ACE+ and Star-CD, in conjunction with custom written user-defined subroutines. In these commercial CFD packages, the upwind finite volume method and the numerical algorithm of semi-implicit method for pressure linked equations (SIMPLE) play the crucial role. Unusually, a finite element version of semi-implicit projection scheme is presented in [35, 36] for a unsteady single-phase transport PEMFC model, where the temperature and two-phase effects and charge transports are neglected in the model. They claimed that this method generally requires much less computer storage and CPU time than the conventional finite element methods, which, however, are not justified in their paper.

It is well known that, the aforementioned commercial CFD packages possess a black-box structure, no kernel code is displayed to the user except the restricted input and output user-defined functions. Provided that their built-in numerical algorithms and computer program present any numerical instability, e.g., an oscillating iteration or a singular solution, few adjustment could be done by the user for the precomposed PDE model and corresponding numerical method. It is even difficult to make any change to the built-in numerical discretization and algorithms because of the unreachable kernel code. In addition, the user interface for importing problem-dependent coefficients/parameters through input panel and/or user defined functions (UDF) is also limited for some complex partial differential equations (PDEs). These essential weaknesses of a black-box type commercial software greatly degrade the power they can provide for a fuel cell simulation. On the other hand, the numerical discretizations/algorithms of fuel cell model need to be frequently updated for the promising fuel cell technology in order to catch up the step of fuel cell's experimental/modeling study and provide an efficient simulation tool for it. For example, the M^2 model bears a discontinuous and degenerate diffusivity in water concentration equation, resulting in an oscillating nonlinear iteration if using the standard finite volume method provided by commercial CFD packages [49, 50]. Such instable computation could lead the entire nonlinear iteration up to tens of thousands of steps or even nonconvergent iteration, making the whole computation inaccurate and

even highly expensive in multidimensional case.

To attack such a nonlinear, discontinuous and degenerate water diffusivity arising from M^2 model, the numerical technique of Kirchhoff and its inverse transformation will be a good candidate. It is well known that the Kirchhoff transformation has been widely used for the nonlinear thermal diffusion problem with temperature dependent properties [9, 13, 31, 38, 41, 58], the global pressure formulation of two-phase subsurface flow in porous media [10, 44], and the nonlinear, degenerate water flow equation (Richards' equation) in variably saturated porous media [4,5,10,17,18,28,29,37,40,42,43,52–54,57]. By means of an appropriate integral transformation, it essentially converts the nonlinear diffusion term into a linear one, making the linearization unnecessary in many cases, and eliminate the degeneracy and discontinuity from the original problem. Thus, such reformulation due to Kirchhoff transformation may make the entire nonlinear iteration of PEMFC simulation efficient and robust.

Aiming at the numerical difficulties arising from multiphase PEMFC simulation, Kirchhoff transformation technique was first introduced to a 2D simplified two-phase transport PEMFC model in [49,50], where the nonlinear water equation bearing with discontinuous and degenerate diffusivity is reformulated to a semilinear Poisson equation in terms of Kirchhoff transformation, and the nonlinear iteration is significantly sped up for a 2D M^2 model in the cathode of PEMFC by means of a well designed in-house finite element code. Nevertheless, the PEMFC model therein is greatly simplified by considering mass, momentum and water species equations on the cathode side only.

Therefore, in this paper we are dedicated in introducing Kirchhoff transformation technique to a three-dimensional, multicomponent, two-phase, multiphysics PEMFC model, furthermore, implementing the entire numerical simulation with a novel automated finite element/finite volume program generator (FEPG) [1, 21, 59, 66] in a flexible, efficient and self-contained fashion. To this end, we introduce a high-level computer programming language-algorithm description language, and its key role of generating source code for finite element and finite volume methods. Together with component programming and formula library techniques, FEPG can rapidly generate source code of finite element/finite volume method for a multiphysics problem by writing and compiling a script. In this script, in terms of the algorithm description language, one can use familiar concepts/descriptions to specify the original PDEs' weak forms, the involved finite element/finite volume formulae, and the algorithms of discretization/linearization as a problem-dependent part for a PDE problem, and let FEPG system worry about the problem-independent part of the remaining programs. With the help of FEPG, one can focus on the efficient algorithm research for the numerical method of PEMFC model without being distracted by the tedious computer programming on finite element/finite volume methods. Numerical accomplishments made in this paper will demonstrate that the combination of the efficient numerical techniques and the powerful FEPG system provides a potent simulation tool for PEMFC modeling and simulations.

The rest of this paper is organized as follows. We introduce the 3D multicomponent, two-phase, multiphysics PEMFC model in Section 2 and the efficient numerical

techniques in Section 3. In Section 4, we demonstrate the FEPG script files for finite element/finite volume method of PEMFC model. The numerical simulation using the generated source code from FEPG is carried out and the obtained solutions are elucidated in Section 5.

2 A two-phase, multiphysics model of PEMFC

Let us address a steady state two-phase transport model of PEMFC which consists of the conservations of momentum, mass, species, charges and energy, and defines in the gas channels, gas diffusion (backing) layers (GDLs), catalyst layers (CLs), current collectors at both anode and cathode and the membrane in between, as schematically shown in Fig. 1. In the following subsection we will present the governing equations of this multicomponent, two-phase, multiphysics model for an entire PEMFC [48, 62], the principle unknowns of which include the mixture velocity $\vec{u}$, pressure p, species molar concentrations C^{H_2O},C^{H_2},C^{O_2}, proton potential Φ_e and electron potential Φ_s. The definitions of the involved various physical coefficients and parameters can be referred to [32,48,60,62,63].

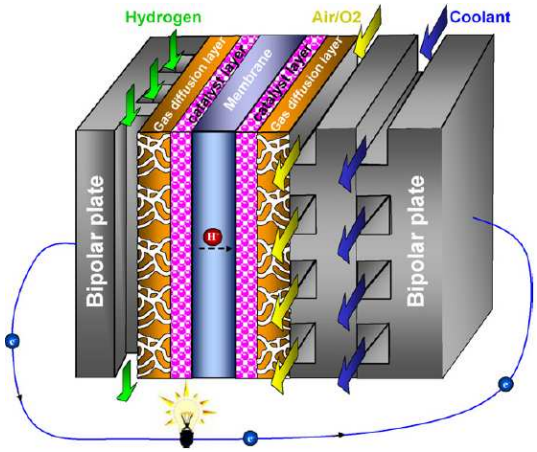


Figure 1: Schematic domain of a 3D PEMFC [62].

2.1 Governing equations

To concentrate on the most important multiphase feature, different from the one given in [48], in this paper we treat the two-phase transport model of PEMFC as an isotropic, isothermal one, and we neglect the water back diffusion effect through membrane in order to focus on the main goal of this paper. A more complete PEMFC model can be referred to [48, 62]. Thus, the governing equations of PEMFC are read as

Mass equation

$$\nabla\cdot(\rho\vec{u}) = S_m. \tag{2.1}$$

Momentum equation

$$\frac{1}{\varepsilon^2}\nabla\cdot(\rho\vec{u}\vec{u})=-\nabla p+\nabla\cdot(\mu\nabla\vec{u})+S_p. \tag{2.2}$$

Note that (2.1) and (2.2) are defined everywhere in PEMFC except the membrane.

Water concentration equation

$$\begin{cases} \nabla\cdot(\gamma_c\vec{u}C^{H_2O})=\nabla\cdot\big(\Gamma(C^{H_2O})\nabla C^{H_2O}\big)+G_{H_2O}+S_{H_2O}, & \text{porous media} \quad (a),\\ \nabla\cdot(\vec{u}C^{H_2O})=\nabla\cdot(D_g^{H_2O}\nabla C^{H_2O}), & \text{gas channels} \quad (b), \end{cases} \tag{2.3}$$

where the water diffusivity $\Gamma(C^{H_2O})$ in porous media is defined as

$$\Gamma(C^{H_2O})=\begin{cases} \Big(\dfrac{1}{M^{H_2O}}-\dfrac{C_{\text{sat}}}{\rho_g}\Big)\Gamma_{\text{capdiff}}, & \text{if } C^{H_2O}\geq C_{\text{sat}} \text{ or } s>0,\\ f(\varepsilon)D_g^{H_2O}, & \text{if } C^{H_2O}<C_{\text{sat}} \text{ or } s=0, \end{cases} \tag{2.4}$$

here Γ_{capdiff} denotes the capillary diffusivity, given by

$$\Gamma_{\text{capdiff}}=\frac{M^{H_2O}}{\rho_l-C_{\text{sat}}M^{H_2O}}\frac{\lambda_l\lambda_g}{\nu}\sigma\cos(\theta_c)(K\varepsilon)^{\frac{1}{2}}\frac{dJ(s)}{ds}, \tag{2.5a}$$

$$D_g^{H_2O}=\begin{cases} \dfrac{1.1028\times10^{-4}}{P_a}\Big(\dfrac{T}{353.15}\Big)^{1.5}, & \text{anode},\\ \dfrac{3.89\times10^{-5}}{P_c}\Big(\dfrac{T}{353.15}\Big)^{1.5}, & \text{cathode}, \end{cases} \tag{2.5b}$$

$$g(s)=(1-s)^{1.5},\quad f(\varepsilon)=\begin{cases} \varepsilon\Big(\dfrac{\varepsilon-\varepsilon_p}{1-\varepsilon_p}\Big)^{0.521}, & \text{gas diffusion layers},\\ \varepsilon^{1.5}, & \text{catalyst layers}. \end{cases} \tag{2.5c}$$

Hence, $f(\varepsilon)=1$ in gas channels where $\varepsilon=1$. γ_c denotes an advection correction factor, defined as

$$\gamma_c=\begin{cases} \dfrac{\rho}{C^{H_2O}}\Big(\dfrac{\lambda_l}{M^{H_2O}}+\dfrac{\lambda_g}{\rho_g}C_{\text{sat}}\Big), & \text{for water},\\ \dfrac{\rho\lambda_g}{\rho_g(1-s)}, & \text{for other species}, \end{cases} \tag{2.6}$$

and $\gamma_c=1$ in gas channels where $s=0$. The gravity-induced source term due to the capillary action, G_{H_2O}, is read as

$$G_{H_2O}=-\nabla\cdot\Big[\Big(\frac{1}{M^{H_2O}}-\frac{C_{\text{sat}}}{\rho_g}\Big)\frac{\lambda_l\lambda_g}{\nu}K(\rho_l-\rho_g)\vec{g}\Big]. \tag{2.7}$$

Here the liquid saturation, s, and $1-s$ denotes the fraction of the open pore space occupied by the liquid and gas phases, respectively, defined as

$$s=\frac{C^{H_2O}-C_{\text{sat}}}{C_l^{H_2O}-C_{\text{sat}}},\quad \text{where } C_l^{H_2O}=\frac{\rho_l}{M^{H_2O}}. \tag{2.8}$$

Note that (2.3) does not include water transport in membrane because water back diffusion effect in membrane is neglected in this paper.

Hydrogen concentration equation

$$\begin{cases} \nabla\cdot\left(\gamma_c\vec{u}C^{H_2}\right)=\nabla\cdot\left(D_g^{H_2,\text{eff}}\nabla C^{H_2}\right)+G_{H_2}+S_{H_2} \\ \qquad\qquad +\nabla\cdot\left(\dfrac{C^{H_2}}{\rho_g}\Gamma_{\text{capdiff}}\nabla C^{H_2O}\right), & \text{anode porous media} \quad (a), \\ \nabla\cdot\left(\vec{u}C^{H_2}\right)=\nabla\cdot\left(D_g^{H_2}\nabla C^{H_2}\right), & \text{anode channel} \qquad (b), \end{cases} \tag{2.9}$$

where

$$G_{H_2}=\nabla\cdot\left[\frac{C^{H_2}}{\rho_g}\frac{\lambda_l\lambda_g}{\nu}K(\rho_l-\rho_g)\vec{g}\right], \tag{2.10a}$$

$$D_g^{H_2}=\frac{1.1028\times10^{-4}}{P_a}\left(\frac{T}{353.15}\right)^{1.5}, \tag{2.10b}$$

$$D_g^{k,\text{eff}}=D_g^k f(\varepsilon)g(s)\quad(k=H_2,O_2). \tag{2.10c}$$

Oxygen concentration equation

$$\begin{cases} \nabla\cdot(\gamma_c\vec{u}C^{O_2})=\nabla\cdot\left(D_g^{O_2,\text{eff}}\nabla C^{O_2}\right)+G_{O_2}+S_{O_2} \\ \qquad\qquad +\nabla\cdot\left(\dfrac{C^{O_2}}{\rho_g}\Gamma_{\text{capdiff}}\nabla C^{H_2O}\right), & \text{cathode porous media} \quad (a), \\ \nabla\cdot\left(\vec{u}C^{O_2}\right)=\nabla\cdot\left(D_g^{O_2}\nabla C^{O_2}\right), & \text{cathode channel} \qquad (b), \end{cases} \tag{2.11}$$

where

$$G_{O_2}=\nabla\cdot\left[\frac{C^{O_2}}{\rho_g}\frac{\lambda_l\lambda_g}{\nu}K(\rho_l-\rho_g)\vec{g}\right],\quad D_g^{O_2}=\frac{3.2348\times10^{-5}}{P_c}\left(\frac{T}{353.15}\right)^{1.5}. \tag{2.12}$$

Proton potential equation

$$\nabla\cdot(\kappa^{\text{eff}}\nabla\Phi_e)+S_{\Phi_e}=0, \tag{2.13}$$

which is defined in membrane electrode assembly (MEA), the combination of membrane and catalyst layers, where $\kappa_{\text{CL}}^{\text{eff}}=\varepsilon_{mc}^{1.5}\kappa_{\text{mem}}^{\text{eff}}$.

Electron potential equation

$$\nabla\cdot\left[\sigma^{\text{eff}}\nabla\Phi_s\right]+S_{\Phi_s}=0, \tag{2.14}$$

which is defined everywhere in PEMFC except the gas channels.

Source terms In the right hand sides of (2.1)-(2.14), S_i ($i=m,p,H_2O,H_2,O_2,\Phi_e,\Phi_s$) represents the source term of each governing equation, defined as

$$S_m=\begin{cases} -M^{H_2}\dfrac{j_a}{2F}-M^{H_2O}\nabla\cdot\Big(\dfrac{n_d}{F}\vec{i}_e\Big), & \text{anode CL,}\\ M^{O_2}\dfrac{j_c}{4F}-M^{H_2O}\Big[\dfrac{j_c}{2F}+\nabla\cdot\Big(\dfrac{n_d}{F}\vec{i}_e\Big)\Big], & \text{cathode CL,}\end{cases} \tag{2.15a}$$

$$S_p=-\frac{\mu}{K}\vec{u}, \quad \text{porous media,} \tag{2.15b}$$

$$S_{H_2O}=\begin{cases} -\nabla\cdot\Big(\dfrac{n_d}{F}\vec{i}_e\Big), & \text{anode CL,}\\ -\dfrac{j_c}{2F}-\nabla\cdot\Big(\dfrac{n_d}{F}\vec{i}_e\Big), & \text{cathode CL,}\end{cases} \tag{2.15c}$$

$$S_{H_2}=-\frac{j_a}{2F}, \quad \text{anode CL,} \tag{2.15d}$$

$$S_{O_2}=\frac{j_c}{4F}, \quad \text{cathode CL,} \tag{2.15e}$$

$$S_{\Phi_e}=\begin{cases} j_a, & \text{anode CL,}\\ j_c, & \text{cathode CL,}\end{cases} \qquad S_{\Phi_s}=\begin{cases} -j_a, & \text{anode CL,}\\ -j_c, & \text{cathode CL,}\end{cases} \tag{2.15f}$$

where, the volumetric transfer current density of the reaction, j (A/m^3), given by the modified Butler-Volmer equation in the anode and cathode, respectively, as

$$j_a=ai_{0,a}\Big(\frac{C_{H_2}}{C_{H_2,\text{ref}}}\Big)^{\frac{1}{2}}\Big(\frac{\alpha_a+\alpha_c}{RT}F\eta\Big), \tag{2.16a}$$

$$j_c=-ai_{0,c}\exp\Big[-16456\Big(\frac{1}{T}-\frac{1}{353.15}\Big)\Big]\frac{C_{O_2}}{C_{O_2,\text{ref}}}\exp\Big(-\frac{\alpha_c F}{RT}\eta\Big). \tag{2.16b}$$

The above kinetics expressions are derived from the general Butler-Volmer equation based on the facts that the anode exhibits fast electrokinetics and hence a low surface overpotential to justify a linear kinetic rate equation, namely, the Butler-Volmer equation is linearized on the anode side. Whereas, the cathode has relatively slow kinetics to be adequately described by the Tafel equation. The surface overpotential, η, is defined as

$$\eta=\Phi_s-\Phi_e-U_o. \tag{2.17}$$

U_o is the thermodynamic equilibrium potential of the reaction, read as

$$U_o=\begin{cases} 0, & \text{in anode,}\\ 1.23-0.9\times10^{-3}(T-298.15), & \text{in cathode.}\end{cases} \tag{2.18}$$

In this paper we take a uniform temperature $T_{\text{cell}}=80^\circ$C for the above isothermal PEMFC model. The electro-osmotic drag coefficient, n_d, takes a constant value as well. $\vec{i}_e$ represents the protonic current flux, given by $\vec{i}_e=-\kappa^{\text{eff}}\nabla\Phi_e$.

Due to the definition of Darcy's force S_p in (2.15a) and the small permeability ($K\sim 10^{-12}$) in porous media, (2.1) and (2.2) form a modified Navier-Stokes equation to approximate the gas flow state through both channels and porous media without introducing an interfacial boundary conditions on the interface of channel and GDL, i.e., a unified equation in a single domain using a location-dependent parameter K in source term: infinity in channels and small permeability in porous media.

Two-phase mixture relations The two-phase mixture parameters that are involved in the above governing equations are listed in Table 1 [61].

Table 1: Two-phase relations.

Density	$\rho=\rho_l s+\rho_g(1-s)$
Molar Concentration	$C=C_l s+C_g(1-s)$
Velocity	$\rho\vec{u}=\rho_l\vec{u}_l+\rho_g\vec{u}_g$
Kinematic viscosity	$\nu=\left(\frac{k_{rl}}{\nu_l}+\frac{k_{rg}}{\nu_g}\right)^{-1},\ \nu_l=\frac{\mu_l}{\rho_l},\ \nu_g=\frac{\mu_g}{\rho_g}$
Effective viscosity	$\mu=\rho\nu=(\rho_l\cdot s+\rho_g\cdot(1-s))\left(\frac{k_{rl}}{\nu_l}+\frac{k_{rg}}{\nu_g}\right)^{-1}$
Diffusivity	$\rho D^k=\rho_l s D_l^k+\rho_g(1-s)D_g^k$
Advection	$\gamma=\rho(\lambda_l c_l^k/\rho_l+\lambda_g c_g^k/\rho_g)/C$
Mobilities	$\lambda_l=\frac{k_{rl}}{\nu_l}\nu,\lambda_g=\frac{k_{rg}}{\nu_g}\nu,\lambda_l+\lambda_g=1$
Relative permeabilities	$k_{rl}=s^3,k_{rg}=(1-s)^3$

2.2 Computational domain and boundary conditions

As shown in Fig. 2, the domain of a 3D PEMFC with single channel to be simulated is divided into nine subregions in through-plane direction, i.e., along x axis: the anode bipolar plate (current collector) (ABP), anode channel (AGC), anode backing layer (ABL), anode catalyst layer (ACL), membrane (MEM), cathode catalyst layer (CCL), cathode backing layer (CBL), cathode channel (CGC) and cathode bipolar plate (CBP). The domain dimensions and operating parameters of PEMFC are listed in Table 2. The boundary conditions of the governing equations (2.1)-(2.14) are described as follows [48].

Inlet boundaries The inlet velocity $\vec{u}_{\text{in}}$ in a gas channel is expressed by the respective stoichiometric flow ratio, i.e., ξ_a or ξ_c, defined at a reference current density, I_{ref}, as

$$\vec{u}_{\text{in},a}=\frac{I_{\text{ref}}A}{2FC_{\text{in}}^{H_2}A_{\text{in},a}}\xi_a,\qquad \vec{u}_{\text{in},c}=\frac{I_{\text{ref}}A}{4FC_{\text{in}}^{O_2}A_{\text{in},c}}\xi_c. \tag{2.19}$$

Species molar concentrations at the inlet are specified as $C^k=C_{\text{in}}^k$ $(k=H_2O,H_2,O_2)$, which can be determined by the ideal gas law, partial pressure and relative humidity (RH) [48].

Outlet boundaries Fully developed or no-flux conditions are applied to flow, species concentrations and charge potentials at the outlet, given as

$$(pI-\mu\nabla\vec{u})\cdot\vec{n}=0,\quad \frac{\partial C^k}{\partial n}=0,\quad \frac{\partial\Phi_e}{\partial n}=0,\quad \frac{\partial\Phi_s}{\partial n}=0. \tag{2.20}$$

Table 2: Physical parameters and properties.

Parameters/properties	*Symbol*	*Value*	*Unit*
Modeling domain dimensions			
ABP thickness	δ_{ABP}	5×10^{-4}	m
AGC thickness	δ_{AGC}	10^{-3}	m
ABL thickness	δ_{ABL}	3×10^{-4}	m
ACL thickness	δ_{ACL}	10^{-5}	m
MEM thickness	δ_{MEM}	5×10^{-5}	m
CCL thickness	δ_{CCL}	10^{-5}	m
CBL thickness	δ_{CBL}	3×10^{-4}	m
CGC thickness	δ_{CGC}	10^{-3}	m
CBP thickness	δ_{CBP}	5×10^{-4}	m
Cell length	l_{cell}	2.5×10^{-2}	m
Cell width	w_{cell}	2×10^{-3}	m
Channel width	w_{channel}	10^{-3}	m
Operating parameters			
Anode reference current density	$a_0 i_{0,a}^{\text{ref}}$	1.0×10^{9}	A/m^3
Cathode reference current density	$a_0 i_{0,c}^{\text{ref}}$	2.0×10^{4}	A/m^3
Anode/cathode pressures	P	1.0/1.0	atm
Stoichiometry at 0.2A/cm^2	ξ_a/ξ_c	3.0/3.0	
Ambient temperature	T_0	353.15	K
Relative humidity at anode/cathode	RH	100/100%	
Porosity of GDLs/CLs	ε	0.6/0.4	
Percolation threshold	ε_p	0.11	
Dynamic viscosity of liquid water	μ_l	3.56×10^{-4}	kg/m s
Dynamic viscosity of anode gas	μ_a	1.101×10^{-5}	kg/m s
Dynamic viscosity of cathode gas	μ_c	1.881×10^{-5}	kg/m s
Kinematic liquid water viscosity	ν_l	3.533×10^{-7}	m^2/s
Kinematic water vapor viscosity	ν_g	3.59×10^{-5}	m^2/s
Surface tension	σ	0.0625	N/m
Contact angle	θ_c	110°	
Permeability	K	8.69×10^{-12}	m^2
Protonic conductivity of MEM	$\kappa_{\text{mem}}^{\text{eff}}$	10	S/m
Volume fraction of membrane in CLs	ε_{mc}	0.26	
Electronic conductivity in GDL	$\sigma_{\text{GDL}}^{\text{eff}}$	1250	S/m
Electronic conductivity in BPs	$\sigma_{\text{BP}}^{\text{eff}}$	2×10^{4}	S/m
Water molecular weight	M^{H_2O}	0.018015	kg/mol
Hydrogen molecular weight	M^{H_2}	0.00201594	kg/mol
Oxygen molecular weight	M^{O_2}	0.031999	kg/mol
Water Vapor density	ρ_g	0.882	kg/m^3
Liquid water density	ρ_l	971.8	kg/m^3
Electro-osmosis coefficient	n_d	2.5	$\text{H}_2\text{O/H}^+$
Faraday constant	F	96487	A s/mol
Universal gas constant	R	8.314	J/mol/K

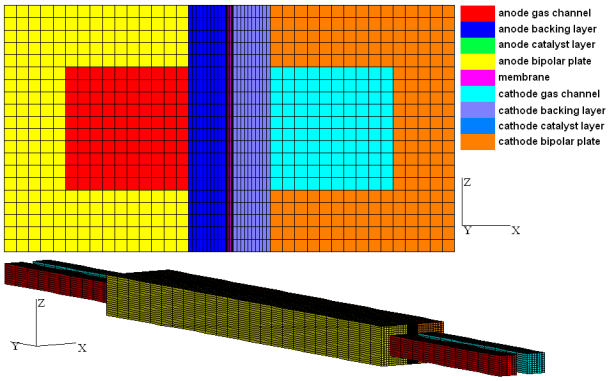


Figure 2: Computational domain and mesh of a single-channel PEMFC.

Walls No-slip and impermeable velocity and no-flux conditions are applied as

$$\vec{u}=0, \quad \frac{\partial p}{\partial n}=0, \quad \frac{\partial C^k}{\partial n}=0 \quad \text{and} \quad \frac{\partial \Phi_e}{\partial n}=0. \tag{2.21}$$

The boundary conditions for electronic phase potential, Φ_s, at the outer surfaces of bipolar plate can be expressed as

$$\begin{cases} \Phi_s=0, & \text{ABP}, \\ \dfrac{\partial \Phi_s}{\partial n}=-\dfrac{I_{\text{ref}}A}{\sigma^{\text{eff}}A_{\text{c,wall}}}, & \text{CBP}, \\ \dfrac{\partial \Phi_s}{\partial n}=0, & \text{otherwise}, \end{cases} \tag{2.22}$$

where $A_{\text{c,wall}}$ is the area of the cathode outer surface.

3 Numerical methods

Similar to the numerical simulations for a 2D simplified two-phase model in the cathode of PEMFC [49,50], the standard finite element or finite volume discretizations can result in an oscillating or even nonconvergent nonlinear iteration for the governing equations of PEMFC (2.1)-(2.14), as shown in the right of Fig. 5. Abundant numerical experiments demonstrate that such instable iteration is primarily due to the discontinuous and degenerate water diffusivity function $\Gamma(C^{H_2O})$ arising from the coexistence of single- and two-phase regions, which can be fairly understood in Fig. 3. Here, $C_{\text{sat}}=16.11\text{mol/m}^3$ is the critical point at which the discontinuity and degeneracy of $\Gamma(C^{H_2O})$ concur and the gaseous water begin to condense to liquid water when $T_{\text{cell}}=80^\circ\text{C}$.

In [49,50], an efficient numerical technique, Kirchhoff transformation, plays a key role in overcoming the discontinuous and degenerate water diffusivity and accelerating nonlinear iteration for a 2D simplified two-phase model defined in the cathode of PEMFC. In

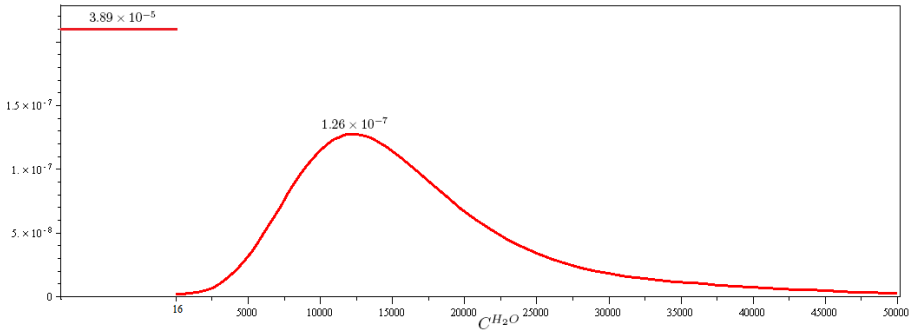


Figure 3: Water diffusivity $\Gamma(C^{H_2O})$ at 80°C.

this section, we will appropriately apply the same numerical technique to a more sophisticated 3D two-phase, multiphysics model for an entire PEMFC, and derive a systematic numerical methodology for dealing with two-phase transport equation arising in PEMFC modeling.

We define a new variable W in porous media in terms of Kirchhoff transformation as follows

$$W(C^{H_2O}) = \int_0^{C^{H_2O}} \Gamma(\omega) d\omega, \tag{3.1}$$

where the integrant function Γ is water diffusivity defined in (2.4). Hence W is a function of water concentration C^{H_2O}. Considering Γ is a single-variable function with respect to C^{H_2O} if $C^{H_2O} \geq C_{\text{sat}}$ or $s > 0$ or constant if $C^{H_2O} < C_{\text{sat}}$ or $s = 0$, due to the fundamental theorem of calculus, the differentiation on both sides of (3.1) leads to

$$\nabla W = \Gamma(C^{H_2O}) \nabla C^{H_2O}. \tag{3.2}$$

Furthermore,

$$\triangle W = \nabla \cdot (\Gamma(C^{H_2O}) \nabla C^{H_2O}), \quad \text{in porous media.}$$

Notice however that for the water concentration equation (2.3) in a single-phase region no matter whether the region is porous media or gas channels, i.e., if $C^{H_2O} < C_{\text{sat}}$ or $s=0$, the diffusivity $\Gamma(C^{H_2O}) = f(\varepsilon) D_g^{H_2O}$ becomes constant, where $f(\varepsilon)=1$ in gas channels. Thus the corresponding Kirchhoff transformation turns to

$$W(C^{H_2O}) = \int_0^{C^{H_2O}} f(\varepsilon) D_g^{H_2O} d\omega = f(\varepsilon) D_g^{H_2O} C^{H_2O}, \tag{3.3}$$

showing that the new variable W becomes a linear function of C^{H_2O} in single-phase region, or inversely,

$$C^{H_2O} = (f(\varepsilon) D_g^{H_2O})^{-1} W, \quad \text{in single-phase region.} \tag{3.4}$$

Therefore, water concentration equation (2.3) can be reformulated as follows in terms of Kirchhoff transformation (3.1)

$$\begin{cases} -\triangle W = -\nabla\cdot(\gamma_c \vec{u} C^{H_2O}) + G_{H_2O} + S_{H_2O}, & \text{in porous media} \quad (a), \\ -\triangle W = -\nabla\cdot\left(\vec{u}(D_g^{H_2O})^{-1} W\right), & \text{in gas channels} \quad (b), \\ W = D_g^{H_2O} C_{\text{in}}^{H_2O}, & \text{at the inlet} \quad (c), \\ \dfrac{\partial W}{\partial n} = 0, & \text{elsewhere on the boundary} \quad (d). \end{cases} \tag{3.5}$$

We observe that only a linear Laplacian term with respect to unknown W stays on the left hand side of the new water equation (3.5), while the original discontinuous and degenerate water diffusivity Γ is absorbed into the Kirchhoff transformation (3.1). Thus, there is only one nonlinearity remaining on the right hand side of (3.5a) where the convection term contains $C^{H_2O}(W)$, an implicit function of W via the inverse Kirchhoff transformation.

Apparently, (3.5) significantly reduces the nonlinearity of original water equation (2.3) to a semilinear problem and makes fast convergence possible for the corresponding nonlinear iteration. Picard's linearization method is sufficient to obtain a fast convergence for this semilinear equation, in which C^{H_2O} is updated from W by inverse Kirchhoff transformation at each iteration step. In contrast to (3.5a), (3.5b) is just a linear convection-diffusion equation in gas channels. Due to the large velocity $\vec{u}$ therein, the convection term of (3.5b) is actually dominant. Therefore, we cannot treat this dominant convection term as an additional source term. Instead, we reformulate it to an explicit convection form with respect to W via linear Kirchhoff transformation (3.4) in (3.5b). To avoid numerical instability arising from this dominant convection term, we may adopt upwind scheme to stabilize the numerical computation and produce a smooth solution.

Thus, in view of the weak nonlinearity in (3.5), we can expect a fast convergence for W, and further for C^{H_2O}, provided that an accurate and efficient method can be designed to implement the inverse Kirchhoff transformation of (3.1), and a upwind scheme can be applied to finite element discretization without loss of the existing advantages of finite element method.

There exists no explicit expression for the inverse function of Kirchhoff transformation (3.1). It is actually nontrivial to compute the inverse Kirchhoff transformation directly. The simple Look-Up Table (LUT) method has been criticized in [49, 50] due to its expensive computation in finding W. Considering $W(C^{H_2O})$ is a strictly increasing continuous function of C^{H_2O}, Newton's method turns out to be an efficient approach to carry out the inverse Kirchhoff transformation. One can rewrite the Kirchhoff transformation as follows

$$W = W_s + W_d(C^{H_2O}), \tag{3.6}$$

where

$$W_s = \int_0^{C_{\text{sat}}} f(\varepsilon) D_g^{H_2O} d\omega = f(\varepsilon) D_g^{H_2O} C_{\text{sat}}, \qquad W_d(C^{H_2O}) = \int_{C_{\text{sat}}}^{C^{H_2O}} \Gamma(\omega) d\omega.$$

Hence, if $C^{H_2O}\leq C_{\text{sat}}$ or $W\leq W_s$, then W is just a simple linear function of C^{H_2O} by (3.3), and (3.4) is sufficient to get the desired C^{H_2O}. The nontrivial case is when $C^{H_2O}>C_{\text{sat}}$ or $W>W_s$ in which $\Gamma(\omega)$ is actually a third degree polynomial of ω. Note that the definite integral of $\Gamma(\omega)$ can be explicitly evaluated, and $(W_d(C^{H_2O}))'=\Gamma(C^{H_2O})$ by fundamental theorem of calculus, thus the inverse Kirchhoff transformation is able to be dealt with by applying Newton's method to (3.6) for any given $W>W_s$, read as

$$C_{k+1}^{H_2O}=C_k^{H_2O}+\frac{W-W_s-W_d(C_k^{H_2O})}{F(C_k^{H_2O})},\quad k=0,1,2,\cdots. \tag{3.7}$$

This scheme shall converge provably in very few steps with a proper initial guess $C_0^{H_2O}$, based on the local quadratic convergence theory of Newton's method.

To combine the advantages of both upwind scheme and finite element method and conquer the dominant convection in the framework of finite element approach, we employ a combined finite element-upwind finite volume method [14, 47, 51], together with Newton's linearization scheme, to discretize the governing equations of PEMFC (2.1), (2.2), (3.5), (2.9)-(2.14), where the water concentration equation (2.3) is replaced by (3.5), and the source terms (2.15c) and (2.15f) are linearized by Newton's method. To save the entire computational cost and reduce the workload of the linear algebraic solver later on, we adopt a Gauss-Seidel type decoupling approach to decouple the multiphysics PEMFC problem in the following sequence.

We first define a piecewise linear finite element space S_h on the hexahedral triangulation of domain shown in Fig. 2. Provided $(\Phi_{e,h}^n,\ \Phi_{s,h}^n,\ \vec{u}_h^n,\ p_h^n,\ C_h^{H_2O,n},\ C_h^{H_2,n},\ C_h^{O_2,n})$ is given, find $(\Phi_{e,h}^{n+1},\ \Phi_{s,h}^{n+1},\ \vec{u}_h^{n+1},\ p_h^{n+1},\ W_h^{n+1},\ C_h^{H_2,n+1},\ C_h^{O_2,n+1})\in S_h$ such that for any $(\widetilde{\Phi_e},\ \widetilde{\Phi_s},\ \widetilde{\vec{u}},\ \widetilde{p},\ \widetilde{W},\ \widetilde{C^{H_2}},\ \widetilde{C^{O_2}})\in S_h$, the following discretizations of seven governing equations hold ($n=0,1,2,\cdots$):

1. FEM discretization of proton potential equation

$$\left(\kappa^{\text{eff}}\nabla\Phi_{e,h}^{n+1},\nabla\widetilde{\Phi_e}\right)-\left(\frac{\partial S_{\Phi_{e,h}}^n}{\partial\Phi_{e,h}}\Phi_{e,h}^{n+1},\widetilde{\Phi_e}\right)=\left(S_{\Phi_{e,h}}^n,\widetilde{\Phi_e}\right)-\left(\frac{\partial S_{\Phi_{e,h}}^n}{\partial\Phi_{e,h}}\Phi_{e,h}^n,\widetilde{\Phi_e}\right). \tag{3.8}$$

2. FEM discretization of electron potential equation

$$\left(\sigma_s^{\text{eff}}\nabla\Phi_{s,h}^{n+1},\nabla\widetilde{\Phi_s}\right)-\left(\frac{\partial S_{\Phi_{s,h}}^n}{\partial\Phi_{s,h}}\Phi_{s,h}^{n+1},\widetilde{\Phi_s}\right)=\left(S_{\Phi_{s,h}}^n,\widetilde{\Phi_s}\right)-\left(\frac{\partial S_{\Phi_{s,h}}^n}{\partial\Phi_{s,h}}\Phi_{s,h}^n,\widetilde{\Phi_s}\right), \tag{3.9}$$

where we use the updated $\Phi_{e,h}^{n+1}$ from (3.8) to compute $S_{\Phi_{s,h}}^n$ via (2.15f) and (2.16).

3. FEM-Upwind FVM discretization of momentum and continuity equations

$$\begin{cases}\left(\mu^n\nabla\vec{u}_h^{n+1},\nabla\widetilde{\vec{u}}\right)-\left(p_h^{n+1},\nabla\cdot\widetilde{\vec{u}}\right)-\left(S_{p,h}^{n+1},\widetilde{\vec{u}}\right)+\delta(h^2)\left(\nabla p_h^{n+1},\nabla\widetilde{p}\right)\\ \quad+\sum_{i=1}^N\widetilde{\vec{u}}_i\sum_{j\in\Lambda_i}\frac{1}{\varepsilon^2}\int_{\Gamma_{ij}}\left(\rho_h^n\vec{u}_h^n\cdot\vec{n}\right)ds\left(r_{ij}\vec{u}_{h,i}^{n+1}+(1-r_{ij})\vec{u}_{h,j}^{n+1}\right)=0, & (a),\\ \left(\nabla\cdot\vec{u}_h^{n+1},\widetilde{p}\right)=\left(\frac{S_{m,h}^n}{\rho_h^n},\widetilde{p}\right)-\left(\frac{\nabla\rho_h^n}{\rho_h^n}\cdot\vec{u}_h^n,\widetilde{p}\right), & (b),\end{cases} \tag{3.10}$$

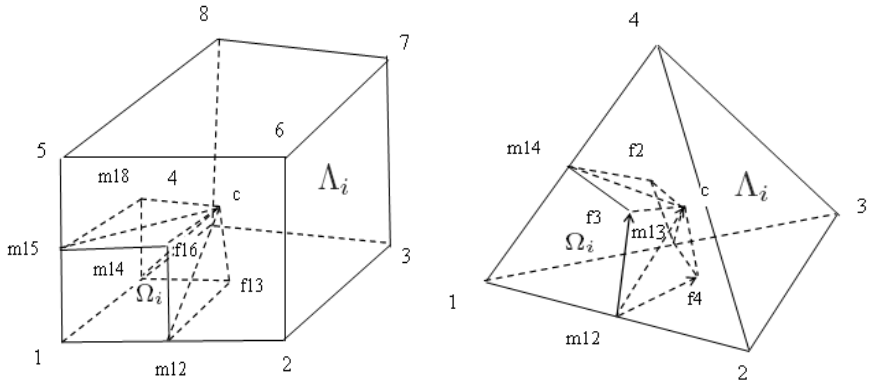


Figure 4: A portion of control volume Ω_i surrounded by dotted faces and cell faces in one cell Λ_i of 3D dual mesh.

where $S_{p,h}^{n+1}=-\mu^n\vec{u}_h^{n+1}/K$ holds in gas diffusion and catalyst layers only, $S_{m,h}^n$ is computed using $\Phi_{e,h}^{n+1}$,$\Phi_{s,h}^{n+1}$,$C_h^{H_2O,n}$,$C_h^{H_2,n}$ and $C_h^{O_2,n}$. Since we employ Q1Q1 conforming finite element to linearly approximate both velocity and pressure in (3.10), a pressure-stabilizing term, $-\delta(h^2)\triangle p_h$, must be added to the momentum equation (2.2) in order to ensure such Q1Q1 mixed finite element is stable [8, 16, 55]. Correspondingly, the weak form term, $\delta(h^2)(\nabla p_h,\nabla\widetilde{p})$, is added to (3.10a) as well. The upwind finite volume discretization term in (3.10a) is derived from the upwind scheme in finite volume approach for the divergent form of convection term in [47, 51], where $F_{ij}=\int_{\Gamma_{ij}}\left(\rho_h^n\vec{u}_h^n\cdot\vec{n}\right)ds$ is the numerical flux and is used to determine the upwind parameter r_{ij} via the following fully upwind scheme

$$r_{ij}=\begin{cases}1, & \text{if } F_{ij}>0,\\ 0, & \text{if } F_{ij}<0,\\ 0.5, & \text{if } F_{ij}=0.\end{cases} \tag{3.11}$$

The notations arising in the upwind-finite volume term in (3.10a) are shown in Fig. 4.

4. FEM-Upwind FVM discretization of water transport

$$\begin{cases}\left(\nabla W_h^{n+1},\nabla\widetilde{W}\right)=\left(\gamma_{c,h}^n\vec{u}_h^{n+1}C_h^{H_2O,n},\nabla\widetilde{W}\right)+\left(G_{H_2O,h}^n,\widetilde{W}\right)\\ \qquad\qquad +\left(S_{H_2O,h}^n,\widetilde{W}\right), & \text{in porous media}\quad (a),\\ \left(\nabla W_h^{n+1},\nabla\widetilde{W}\right)=-\sum_{i=1}^{N}\widetilde{W}_i\sum_{j\in\Lambda_i}\int_{\Gamma_{ij}}\left[(D_{g,h}^{H_2O,n})^{-1}\vec{u}_h^{n+1}\cdot\vec{n}\right]ds\\ \qquad\qquad \left(r_{ij}W_{h,i}^{n+1}+(1-r_{ij})W_{h,j}^{n+1}\right), & \text{in gas channels}\quad (b),\end{cases} \tag{3.12}$$

where $F_{ij}=\int_{\Gamma_{ij}}[(D_{g,h}^{H_2O,n})^{-1}\vec{u}_h^{n+1}\cdot\vec{n}]ds$ is the numerical flux in gas channels of both anode and cathode. The coefficients and source term are computed using $\Phi_{e,h}^{n+1}$, $\Phi_{s,h}^{n+1}$, $C_h^{H_2O,n}$,$C_h^{H_2,n}$ and $C_h^{O_2,n}$.

5. FEM-Upwind FVM discretization of hydrogen transport

$$\begin{cases} \left(D_{g,h}^{H_2,\text{eff},n}\nabla C_h^{H_2,n+1},\nabla\widetilde{C^{H_2}}\right)-\left(\gamma_{c,h}^n\vec{u}_h^{n+1}C_h^{H_2,n+1},\nabla\widetilde{C^{H_2}}\right) \\ -\left(\dfrac{\partial S_{H_2,h}^n}{\partial C_h^{H_2}}C_h^{H_2,n+1},\widetilde{C^{H_2}}\right)=\left(G_{H_2,h}^n,\widetilde{C^{H_2}}\right)+\left(S_{H_2,h}^n,\widetilde{C^{H_2}}\right) \\ -\left(\dfrac{\partial S_{H_2,h}^n}{\partial C_h^{H_2}}C_h^{H_2,n},\widetilde{C^{H_2}}\right)-\left(\dfrac{C_h^{H_2,n}}{\rho_{g,h}}\Gamma_{\text{capdiff},h}^n\nabla C_h^{H_2O,n+1},\nabla\widetilde{C^{H_2}}\right), & \text{anode porous media,} \\ \left(D_{g,h}^{H_2,n}\nabla C_h^{H_2,n+1},\nabla\widetilde{C^{H_2}}\right)+\sum\limits_{i=1}^{N}\widetilde{C_i^{H_2}}\sum\limits_{j\in\Lambda_i}\int_{\Gamma_{ij}}\left[\vec{u}_h^{n+1}\cdot\vec{n}\right]ds \\ \left(r_{ij}C_{h,i}^{H_2,n+1}+(1-r_{ij})C_{h,j}^{H_2,n+1}\right)=0, & \text{anode channel,} \end{cases} \tag{3.13}$$

where the numerical flux $F_{ij}=\int_{\Gamma_{ij}}[\vec{u}_h^{n+1}\cdot\vec{n}]ds$ in anode channel. The coefficients and source term are computed using $\Phi_{e,h}^{n+1}$, $\Phi_{s,h}^{n+1}$, $C_h^{H_2O,n+1}$, $C_h^{H_2,n}$ and $C_h^{O_2,n}$.

6. FEM-Upwind FVM discretization of oxygen transport

$$\begin{cases} \left(D_{g,h}^{O_2,\text{eff},n}\nabla C_h^{O_2,n+1},\nabla\widetilde{C^{O_2}}\right)-\left(\gamma_{c,h}^n\vec{u}_h^{n+1}C_h^{O_2,n+1},\nabla\widetilde{C^{O_2}}\right) \\ -\left(\dfrac{\partial S_{O_2,h}^n}{\partial C_h^{O_2}}C_h^{O_2,n+1},\widetilde{C^{O_2}}\right)=\left(G_{O_2,h}^n,\widetilde{C^{O_2}}\right)+\left(S_{O_2,h}^n,\widetilde{C^{O_2}}\right) \\ -\left(\dfrac{\partial S_{O_2,h}^n}{\partial C_h^{O_2}}C_h^{O_2,n},\widetilde{C^{O_2}}\right)-\left(\dfrac{C_h^{O_2,n}}{\rho_{g,h}}\Gamma_{\text{capdiff},h}^n\nabla C_h^{H_2O,n+1},\nabla\widetilde{C^{O_2}}\right), & \text{cathode porous media,} \\ \left(D_{g,h}^{O_2,n}\nabla C_h^{O_2,n+1},\nabla\widetilde{C^{O_2}}\right)+\sum\limits_{i=1}^{N}\widetilde{C_i^{O_2}}\sum\limits_{j\in\Lambda_i}\int_{\Gamma_{ij}}\left[\vec{u}_h^{n+1}\cdot\vec{n}\right]ds \\ \left(r_{ij}C_{h,i}^{O_2,n+1}+(1-r_{ij})C_{h,j}^{O_2,n+1}\right)=0, & \text{cathode channel,} \end{cases} \tag{3.14}$$

where the numerical flux $F_{ij}=\int_{\Gamma_{ij}}[\vec{u}_h^{n+1}\cdot\vec{n}]ds$ in cathode channel. The coefficients and source term are computed using $\Phi_{e,h}^{n+1}$, $\Phi_{s,h}^{n+1}$, $C_h^{H_2O,n+1}$, $C_h^{H_2,n+1}$ and $C_h^{O_2,n}$.

A careful symbolic calculation shows that all the derivatives of the source terms defined in (2.15c) and (2.15f) are negative, i.e.,

$$\frac{\partial S_{\Phi_{e,h}}^n}{\partial\Phi_{e,h}}<0,\quad \frac{\partial S_{\Phi_{s,h}}^n}{\partial\Phi_{s,h}}<0,\quad \frac{\partial S_{H_2,h}^n}{\partial C_h^{H_2}}<0,\quad \frac{\partial S_{O_2,h}^n}{\partial C_h^{O_2}}<0.$$

Thus, the zero-order terms in the weak formulations, which the above derivatives of source terms arise from, turn out to be positive on the left hand side of (3.8), (3.9), (3.13) and (3.14). On the other hand, these derivatives are all proportional to the transfer current density determined by (2.16) with high magnitude. Hence, these positive zero-order terms may strongly stabilize the entire discretization system and further accelerate the entire nonlinear iteration.

For the initial guesses of Newton's linearizations in the above discretizations, we simply choose uniform initial values for each principle variable in terms of their Dirichlet

boundary conditions, i.e., $\Phi^0_{e,h}=0$, $\Phi^0_{s,,h}=0$ at anode and let $\Phi^0_{s,,h}$ equal to an open-circuit voltage at cathode, e.g., $\Phi^0_{s,,h}=0.4V$, and $\vec{u}^0_h=\vec{u}_{in}$, $C^{H_2O,0}_h=C^{H_2O}_{in}$, $C^{H_2,0}_h=C^{H_2}_{in}$, $C^{O_2,0}_h=C^{O_2}_{in}$.

It is easy to verify that each solution $v^n_h\to v_h\,(v=\Phi_e,\Phi_s,u,p,C^{H_2O},C^{H_2},C^{O_2})$ as $n\to\infty$. Thus, the above Newton's or Picard's linearization-induced discretizations converge to the original finite element-upwind finite volume discretizations without linearization. Whereas, the speed of this convergent process solely depends on the efficient algorithms we employ to design the discretizations, e.g., Kirchhoff transformation, upwind finite volume scheme and Newton's linearization. By means of an appropriately preconditioned GMRES iterative solver [7, 12, 19, 20, 27, 45, 47] at each iteration step, the nonsymmetric positive definite linear algebraic systems, which are derived from the above numerical discretizations, are able to be solved in the manner of efficiency and robustness.

4 Introduction to FEPG

FEPG is the abbreviation of an automated finite element/finite volume program generator. It was first invented for finite element program generation by Guoping Liang, one of the authors of this paper, in 1990, and has been significantly developed in the past two decades, including finite volume program generation as well. FEPG is a self-contained general-purpose software package that can automatically generate finite element/finite volume source code based on a high-level algorithm description language (script), component programming and human intelligence technologies. Aiming at providing user a very flexible and powerful simulation tool to produce finite element/finite volume solutions for multiphysics problems, FEPG can work for many kinds of problems in science and engineering computing that may be described by a complex system of PDEs, spreading all over the fields such as solid mechanics, fluid dynamics, physics, chemistry, thermodynamics, biology, geophysics, astronomy, electromagnetics, meteorology and so on.

A major breakthrough of FEPG is that, it provides user a special script-algorithm description language-to input the definitions of PDEs based on finite element/finite volume method. This is a significant difference from many other finite element program packages (e.g., ANSYS, NASTRAN) that deliver to user a black box for solving problems in some particular fields, which is very difficult for the end-user and the third-party to maintain and develop in order to match their particular needs. Moreover, FEPG is also more extraordinary than other automated program generation packages in virtue of its unique high-level algorithm description language. Although other automated generation packages (e.g., FreeFEM, Comsol, FEniCS, DUNE-FEM, Abaqus) also allow user to define/modify the original PDEs based on their weak forms by either directly coding in C++/Fortran languages or calling the built-in function-library, few of them provide a special script language for user to conveniently define PDEs, finite element/finite volume discretizations and numerical algorithms.

FEniCS [3] does have its own high-level scripts called Unified Form Language (UFL),

together with DOLFIN component and through Python interface, to express weak forms, discretizations and linear algebraic solvers for the finite element method of PDEs. In this sense, FeniCS is similar with FEPG. However, FEniCS requires users to grasp Python programming language (like C and Java) first because Python plays the crucial role on the interface of UFL script and finite element method. Another numerical PDEs' toolbox, Distributed and Unified Numerics Environment (DUNE-FEM) [2], is also a finite element package based on the weak form of PDEs. But DUNE-FEM does not provide a high-level script language for users, C++ programming language is actually the minimum requirement of DUNE that users must possess.

On the other hand, FEPG not only just automatically generates source code for finite element method based on the weak form of PDEs, moreover, it also provides automated code generation for finite volume/difference method according to their nodal equations. Actually, the most extraordinary thing is that, FEPG can even combine finite element method with finite volume method together in one unified framework of automated generation, and efficiently implement the combined finite element-upwind finite volume method for a convection-dominated diffusion reaction problem, as demonstrated in Section 5. Nevertheless, FEniCS does not possess of the automated generation for finite volume/difference method. Although DUNE-FEM supports both finite element method and finite volume/difference method, it still cannot implement the combined finite element-upwind finite volume method in one unified framework for a convection-dominated diffusion problem.

FEniCS employs formula library technique to define finite element space. However, the drawback of using library technique is that, it will be very hard for users to creatively design their new element with its special basis function that actually does not exist in the element library. Distinctively, besides providing formula library for users, FEPG also possesses such power to create a new element by carefully defining its piecewise basis functions in terms of the provided algorithm description script. This is a significant advantage for users to conduct their creative research on finite element method.

Hence, the users of FEniCS, DUNE-FEM and etc. are all required to have professional training on advanced computer programming in C/C++/Python languages. Whereas, FEPG does not have such demand for user, what it needs from user is just a script to describe the weak form of PDEs, and some basic user-defined functions using FORTRAN language, if necessary. On the other hand, many automated program generation packages do not provide source code for user, but FEPG does. FEPG can generate all of plain source codes in FORTRAN for user, which can be transferred to any other platforms for free, then easily compiled and ran without any additional requirement on compiling environment. Thus, based on the FEPG-generated source code, users can quickly create their own programming environment and software development for finite element/finite volume programs, where FEPG offers them with ultimate degree of freedom to investigate up-to-date numerical tricks, such as element types and algorithms. Owning to the above features, FEPG makes it become possible for people to implement and deliver an open, error-free finite element/finite volume source code within days or even hours that

would otherwise take them months or even years.

Roughly speaking, by virtue of the high-level algorithm description language, together with component programming technique provided by FEPG, user can efficiently construct his finite element/finite volume code by writing a series of script files. These scripts are used to define piecewise basis function, numerical quadrature, coordinates transformation, weak form, nodal equation, discretization/linearization scheme, linear algebraic solver, and etc. Each script has its own syntax and input format which are particularly designed for finite element/finite volume method. For example, a simple line in the finite element script

```
[u_i/x_j;u_i/x_j]
```
(4.1)

actually means the following weak form for a 3D Laplacian term $-\triangle\vec{u}$:

$$\begin{aligned}&\Big(\frac{\partial u}{\partial x},\frac{\partial\widetilde{u}}{\partial x}\Big)+\Big(\frac{\partial u}{\partial y},\frac{\partial\widetilde{u}}{\partial y}\Big)+\Big(\frac{\partial u}{\partial z},\frac{\partial\widetilde{u}}{\partial z}\Big)+\Big(\frac{\partial v}{\partial x},\frac{\partial\widetilde{v}}{\partial x}\Big)+\Big(\frac{\partial v}{\partial y},\frac{\partial\widetilde{v}}{\partial y}\Big)+\Big(\frac{\partial v}{\partial z},\frac{\partial\widetilde{v}}{\partial z}\Big)\\&\quad+\Big(\frac{\partial w}{\partial x},\frac{\partial\widetilde{w}}{\partial x}\Big)+\Big(\frac{\partial w}{\partial y},\frac{\partial\widetilde{w}}{\partial y}\Big)+\Big(\frac{\partial w}{\partial z},\frac{\partial\widetilde{w}}{\partial z}\Big),\end{aligned}\tag{4.2}$$

where $\vec{u}=(u,v,w)^T$ is a 3D vector, and $\widetilde{\vec{u}}=(\widetilde{u},\widetilde{v},\widetilde{w})^T$ represents the corresponding test function. The script syntax, e.g., `q/x`, denotes the partial derivative $\partial q/\partial x$, and `[q/x;q/x]` represents an inner product

$$\Big(\frac{\partial q}{\partial x},\frac{\partial\widetilde{q}}{\partial x}\Big)=\int_\Omega\frac{\partial q}{\partial x}\frac{\partial\widetilde{q}}{\partial x}dx.$$

Here, the second q behind the semicolon in the script `[q/x;q/x]` denotes the test function $\widetilde{q}$ with respect to q.

If comparing the script (4.1) with the real weak form (4.2), we can find that (4.1) indeed has a very similar form with the inner product, where the vector-tensor representation is employed in FEPG to simplify the syntax of a vector: a single symbol of an inner product simply represents a sum of nine inner products in the weak form (4.2). Compiled by FEPG-owned script compiler, this one line script (4.1) will immediately turn to a paragraph of Fortran source code which includes all the above weak formulation. More script descriptions can be found in next section where we will apply FEPG to the multiphysics PEMFC model described in Section 2. For the elaborate illustration about FEPG, we refer to [1, 21].

5 FEPG scripts for PEMFC simulation

The algorithm description language (script) plays the crucial role in automatically generating finite element/finite volume source code for a multiphysics PDEs model. It is used to define the weak form of PDEs including discretization and/or linearization, finite element formula including basis function, numerical quadrature and coordinate transform,

and etc. The FEPG system requires user to write a PDE (or VDE if the vector-tensor representation is adopted) file with PDE (or VDE) as its extension name, in which the original PDEs are imported in weak form for finite element discretization, or nodal equation form for finite volume discretization.

In the following we demonstrate how to write this script for the weak formulations (3.8)-(3.14) of a multiphysics PEMFC model defined in Section 2.

To explain distinctly, each line in the script is followed by a comment symbol "//" and then the comment and/or the corresponding weak form term.

Finite element script file: "fuelcell.vde"

```
DEFI                          //Paragraph keyword of name definition, where there is
                              //a beginning keyword for each line as follows.
DISP e s u v w p wh h o       //The name of principle unknowns (Φ_{e,h}^{n+1},Φ_{s,h}^{n+1},u_h^{n+1},
                              //p_h^{n+1},W_h^{n+1},C_h^{H2,n+1},C_h^{O2,n+1}).
COOR x y z                    //The name of coordinate variables.
COEF en sn un vn wn pn        //The known coefficients needed for nonlinear iteration
COEF h2on h2n o2n             //(Φ_{e,h}^n,Φ_{s,h}^n,u_h^n,p_h^n,W_h^n,C_h^{H2,n},C_h^{O2,n}).
FUNC div                      //The name of user-defined divergent function.
SHAP c 8                      //Define nodal basis function with trilinear polynomial.
GAUS c                        //Define Gaussian numerical quadrature with 8 points.
VECT x x y z                  //Define coordinate vector x=(x,y,z)^T.
VECT u u v w                  //Define unknown velocity vector u=(u,v,w)^T.
VECT un un vn wn              //Define known velocity vector u_n=(u_n,v_n,w_n)^T.
$gvs fcfvm                    //Call finite volume code "fcfvm.gvs" to deal with
                              //the dominant advection terms in gas channels.

FUNC                          //Paragraph keyword of user-defined functions.
$c6 include 'declaration'     //User-defined Fortran77 source code to import all
$c6 include 'coefficient'     //necessary coefficients, starting with the keyword $c6
$c6 include 'material'        //that means the first 6 columns are left as blank.
div=+[u_i/x_i]                //Divergent function of velocity ∇·u=∂u/∂x+∂v/∂y+∂w/∂z.

STIF                          //Paragraph keyword of stiffness matrices.
DIST=[e/x_i;e/x_i]*diff_e     //(κ^eff∇Φ_{e,h}^{n+1},∇Φ̃_e).
    -[e;e]*dSe_de             //-(∂S_{Φ_{e,h}}^n/∂Φ_{e,h} Φ_{e,h}^{n+1},Φ̃_e).
    +[s/x_i;s/x_i]*diff_s     //(σ_s^eff∇Φ_{s,h}^{n+1},∇Φ̃_s).
    -[s;s]*dSs_ds             //-(∂S_{Φ_{s,h}}^n/∂Φ_{s,h} Φ_{s,h}^{n+1},Φ̃_s).
    +[u_i/x_j;u_i/x_j]*diff_u //(μ^n∇u_h^{n+1},∇ũ).
    -[u_i;u_i/x_j]*un_j*advect_u //-(ρ/ε^2 u_h^n u_h^{n+1},∇ũ) in porous media.
    +[u_i;u_i]*visc/K         //-(S_{p,h}^{n+1},ũ).
    -[p;div]+[div;p]          //-(p_h^{n+1},∇·ũ)+(∇·u_h^{n+1},p̃).
    +[p/x_i;p/x_i]*det**(2./3.)*C //δ(h^2)(∇p_h^{n+1},∇p̃), where "det" is the
                              //determinant of coordinate transform
                              //matrix and equal to the volume of mesh cell.
```

```
    +[wh/x_i;wh/x_i]          //(∇W_h^{n+1},∇W̃).
    +[h/x_i;h/x_i]*diff_h     //(D_{g,h}^{H2,eff,n} ∇C_h^{H2,n+1},∇C̃^{H2}).
    -[h;h/x_i]*un_i*advect_h  //-(γ_{c,h}^n u_h^n C_h^{H2,n+1},∇C̃^{H2}) in porous media.
    -[h;h]*dSh_dh             //-(∂S_h^{H2,h}/∂C_h^{H2} C_h^{H2,n+1},C̃^{H2}).
    +[o/x_i;o/x_i]*diff_o     //(D_{g,h}^{O2,eff,n} ∇C_h^{O2,n+1},∇C̃^{O2}).
    -[o;o/x_i]*un_i*advect_o  //-(γ_{c,h}^n u_h^n C_h^{O2,n+1},∇C̃^{O2}) in porous media.
    -[o;o]*dSo_do             //-(∂S_h^{O2,h}/∂C_h^{O2} C_h^{O2,n+1},C̃^{O2}).

LOAD=[e]*(Se-dSe_de*en)       //(S_{Φe,h}^n,Φ̃_e)-(∂S_{Φe,h}^n/∂Φ_{e,h} Φ_{e,h}^n,Φ̃_e) in catalyst layers.
    +[s]*(Ss-dSs_ds*sn)       //(S_{Φs,h}^n,Φ̃_s)-(∂S_{Φs,h}^n/∂Φ_{s,h} Φ_{s,h}^n,Φ̃_s) in catalyst layers.
    +[u_i]*0.0+[p]*Sm/rho     //(S_{m,h}^n/ρ_h^n,p̃) in catalyst layers.
    -[p]*un_i*grad_rho_i/rho  //-(∇ρ_h^n/ρ_h^n · u_h^n,p̃).
    +[w]*Sw                   //(S_{H2O,h}^n,W̃) in catalyst layers.
    +[w]*Gw                   //(G_{H2O,h}^n,W̃).
    +[w/x_i]*un_i*wn*advect_w //(γ_{c,h}^n u_h^n C_h^{H2O,n},∇W̃) in porous media.
    +[h]*(Sh-dSh_dh*hn)       //(S_{H2,h}^n,C̃^{H2})-(∂S_{H2,h}^n/∂C_h^{H2} C_h^{H2,n},C̃^{H2}) in ACL.
    +[h]*Gh                   //(G_{H2,h}^n,C̃^{H2}).
    -[h/x_i]*{wn/x_i}*hn*diff_cap //-(C_h^{H2,n}/ρ_{g,h} Γ_{capdiff,h}^n ∇C_h^{H2O,n},∇C̃^{H2}).
    +[o]*(So-dSo_do*on)       //(S_{O2,h}^n,C̃^{O2})-(∂S_{O2,h}^n/∂C_h^{O2} C_h^{O2,n},C̃^{O2}) in CCL.
    +[o]*Go                   //(G_{O2,h}^n,C̃^{O2}).
    -[o/x_i]*{wn/x_i}*on*diff_cap //-(C_h^{O2,n}/ρ_{g,h} Γ_{capdiff,h}^n ∇C_h^{H2O,n},∇C̃^{O2}).

END                           //The end
```

Comparing with the weak forms given in Section 3, we can see that each term in the paragraphs of STIF and LOAD in "fuelcell.vde" exactly matches with the same term in the weak form, as shown in the above comments, where the vector-tensor representation in FEPG plays an important role to simplify the script syntax. In the following finite volume script, the similar trick of alternate subscript is also used to shorten the abundant rotative script writings in the nodal equations. To be more concentrate, we only demonstrate the nodal equation part in the following finite volume script and neglect those definition paragraphs.

Finite volume script file: "fcfvm.fvs"

```
.................
{
&m 12 14 15 23 26 34 37 48 56 58 67 78  //These indices are alternately
```

```
&i 13 18 16 13 27 13 38 18 16 57 27 38  //substituted for the same indices
&j 16 13 18 27 16 38 27 38 57 18 57 57  //used in next paragraph
&r 21 41 51 32 62 43 73 84 65 85 76 87
#
$c6 call squr(xc,x&m,fx&i,fx&j,uc,      //Compute numerical flux, e.g.,:
u&m,fu&i,fu&j,flux&m,flux&mr)           //F_ij = ∫_Γij [u_h^n · n] ds, ∀j ∈ Λ_i.
$c6 call samar(flux&mr,r&m,emu,mode)    //Compute upwind parameter r_ij
$c6 flux&r = -flux&m                    //F_ij + F_ji = 0
$c6 r&r = 1.0-r&m                       //r_ij + r_ji = 1
}
{
&u u
{
&i 1 2 3 4 5 6 7 8
&j 2 3 4 1 8 5 6 7
&k 4 1 2 3 6 7 8 5
&l 5 6 7 8 1 2 3 4

#
EQUATION &i &u                          //Paragraph keyword of nodal equations
[r&i&j*flux&i&j+r&i&k*flux&i&k          //Nodal equation form, e.g.,:
+r&i&l*flux&i&l]u(&i)                   // Σ_{j∈Λ_i} ∫_Γij [u_h^n · n] ds (r_ij C_{h,i}^{H2,n+1}
+[(1-r&i&j)*flux&i&j]u(&j)              //+(1-r_ij) C_{h,j}^{H2,n+1})
+[(1-r&i&k)*flux&i&k]u(&k)
+[(1-r&i&l)*flux&i&l]u(&l)=0
}
}

END                                     //The end
```

Note that the above finite volume script "fcfvm.fvs" will be compiled to "fcfvm.gvs" and further its source code, which will then be called by the element subroutine code generated by the finite element script "fuelcell.vde" due to the line "`$gvs fcfvm`" written in its first definition paragraph.

Via the above scripts, FEPG system integrates the pointwise vertex-centered finite volume nodal equation into the element-wise finite element computation. This ingenious combination can be further interpreted in Fig. 4, where, any grid point shall be encompassed by a control volume which is formed by the adjacent faces between the midpoint of element edges, the center of element faces and the element barycenter. All of such control volumes eventually form a so-called dual mesh, corresponding to the original grid points. Thus, when FEPG conducts finite element computation for local stiffness matrix and local right-hand-side vector in each element, the corresponding part of control volume locating in this element can be computed as well and the obtained nodal equations are contributed to the local finite element stiffness matrix. On the other hand, when all of neighborhood elements of one grid point are swept during the element-wise finite

element computation, an entire control volume, or, all of the finite volume nodal equations associate with this grid point are constructed, simultaneously. A combined finite element-finite volume discretization is then produced, and both advantages of finite element method and finite volume method with upwind scheme are all utilized, sufficiently.

Hence, by writing above finite element/finite volume scripts and then running the automated program generator FEPG, we can immediately obtain an entire finite element/finite volume source code for a complex 3D, two-phase, multiphysics PEMFC model. These generated numerical source codes are easy to maintain and develop since we only need to edit the above finite element/finite volume scripts for a updated PDE model and/or numerical algorithm, where the algorithm description language provides us the tremendous convenience in the sense that it is simply a mathematical language specially working for finite element/finite volume method, and because of this, we are able to claim that FEPG system is capable of dealing with many kinds of physical problems as long as their mathematical PDE model is well defined. On the other hand, FEPG is also an open platform that allows user to develop their own finite element/finite volume schemes as well as numerical algorithms, which in reverse, can enrich FEPG's formula/algorithm libraries as well.

6 Numerical results

By virtue of the automated finite element/finite volume program generator, and implementing the efficient numerical methods designed in Section 3 for the weak formulations (3.8)-(3.14) of the multiphysics PEMFC model defined in Section 2, we are able to obtain a comprehensive efficiency for PEMFC modeling, numerical simulation as well as code development. In this section, we demonstrate this efficiency by elucidating the following attained numerical results for a single-channel PEMFC operating at $I_{\text{ref}}=0.2\text{A/cm}^2$, $P_{\text{a/c}}=1/1\text{atm}$, $T_{\text{cell}}=80^\circ\text{C}$, inlet humidification of $\text{RH}_{\text{a/c}}=100/0\%$, and $\text{Stoich.}_{\text{a/c}}=3/3$, where highly dry air and highly humid and pure hydrogen are fed in the PEMFC. The computer platform we employ to implement the numerical simulation of PEMFC is Dell Precision Workstation T5400 equipped with 2.00GHz Intel Xeon CPU and 8.00 GB of RAM.

The entire numerical simulation is carried out stably and quickly as we expect for a fast iteration. The convergent results are eventually obtained for $(\Phi_{e,h}^{n+1}, \Phi_{s,h}^{n+1}, \vec{u}_h^{n+1}, p_h^{n+1}, C_h^{H_2O,n+1}, C_h^{H_2,n+1}, C_h^{O_2,n+1})$ within 34 nonlinear iteration steps toward a tolerance of relative iteration error, 10^{-6}. The entire convergence history is shown in the left of Fig. 5, in contrast to the nonconvergent history shown in the right which arises from the standard finite element/finite volume approximations without using any specific numerical technique.

The attained numerical results are shown in Figs. 6-12 for the principle unknowns, which are comparable with those of [62] except that a dryout occurs on the anode side of the membrane because the electro-osmotic drag will immediately remove water from

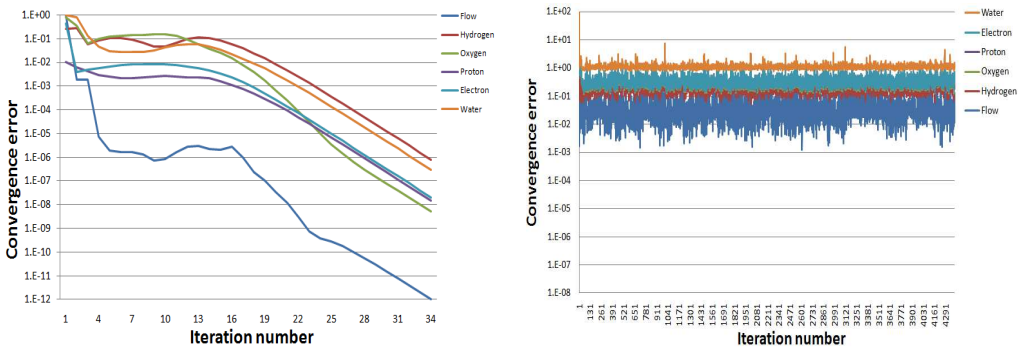


Figure 5: Convergence histories of our FEM-Upwind FVM with Kirchhoff transformation (left) and standard FEM/FVM (right), where the iteration error tolerance is 10^{-6}.

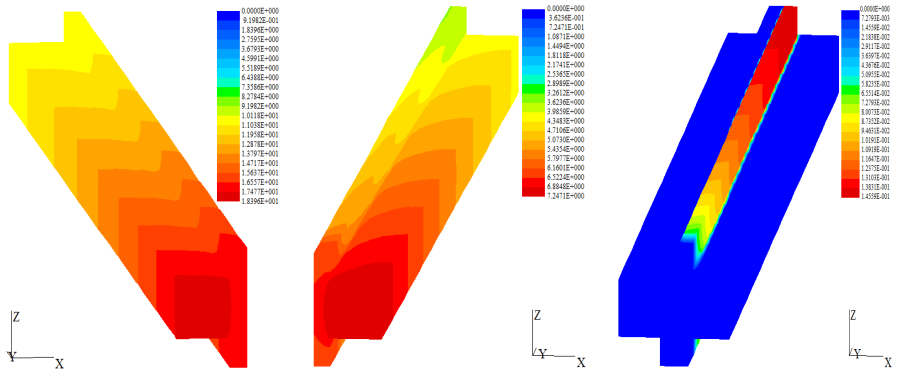


Figure 6: (a) Hydrogen C^{H_2} in anode; (b) Oxygen C^{O_2} in cathode; (c) Liquid water saturation s in both anode and cathode.

the anode side of the membrane, moreover, our model in this paper does not consider water back diffusion effect in membrane. The elucidations of these numerical results are given as follows. The consumption process of hydrogen in anode and oxygen in cathode are displayed in Figs. 6(a), (b), 7 and 8. It can be seen that oxygen and hydrogen concentrations decrease down the channel due to reaction consumption. In contrast to the substantial decrease in the along-channel direction, the concentration only experiences a small decline across the GDLs. In addition, the transport resistance under the land is relatively large, leading to a considerable drop in the reactant concentration. Fig. 6(c), Figs. 9-11 show the contour of the water concentration in both gas channels and porous media. It can be seen that single and slight multiphase regions coexist in PEMFC at the above designated operation- high-humidity in anode and zero-humidity in cathode. The single-phase region is near the inlet of cathode where the dry air is fed in and in the anode gas channel. Due to water production by fuel cell, a small amount of liquid water emerges downstream, i.e., the water concentration $C^{H_2O} > C_{\text{sat}} = 16.11\text{mol/m}^3$ or liquid water saturation $s > 0$ there, and the flow in the diffusion media shifts to gas-

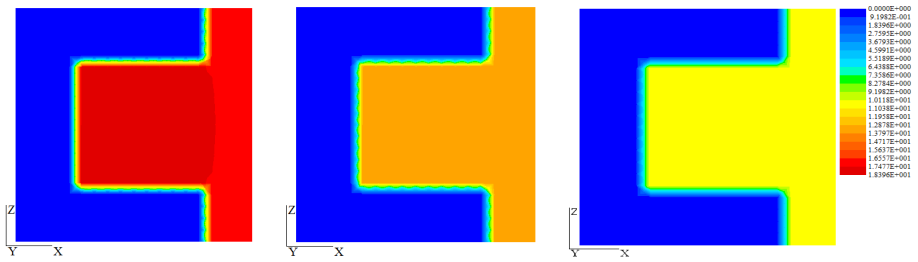


Figure 7: Hydrogen C^{H_2} in XZ-plane (left) near the inlet, (middle) near the central section, (right) near the outlet, in the anode.

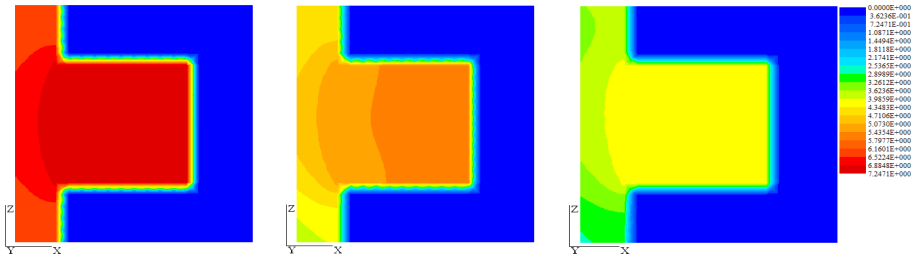


Figure 8: Oxygen C^{O_2} in XZ-plane (left) near the inlet, (middle) near the central section, (right) near the outlet, in the cathode.

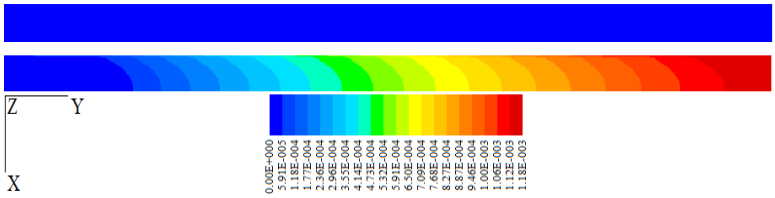


Figure 9: Liquid water saturation s in gas channels of anode (top) and cathode (bottom).

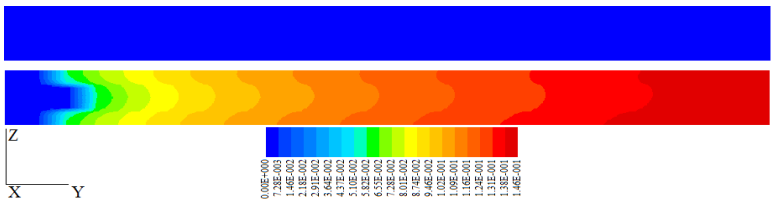


Figure 10: Liquid water saturation s in GDLs of anode (top) and cathode (bottom).

water multiphase flow. Liquid water emerges in the cathode diffusion layer and higher saturation levels appear with the value as high as 15%. However, we can still think there is no liquid water in gas channel since the maximum liquid saturation is only 1.2×10^{-3} in the gas channel of cathode. The anode side of the membrane is mainly dried out or presents the single gas phase because of the effect of the electro-osmotic drag and no water back diffusion is considered in this paper.

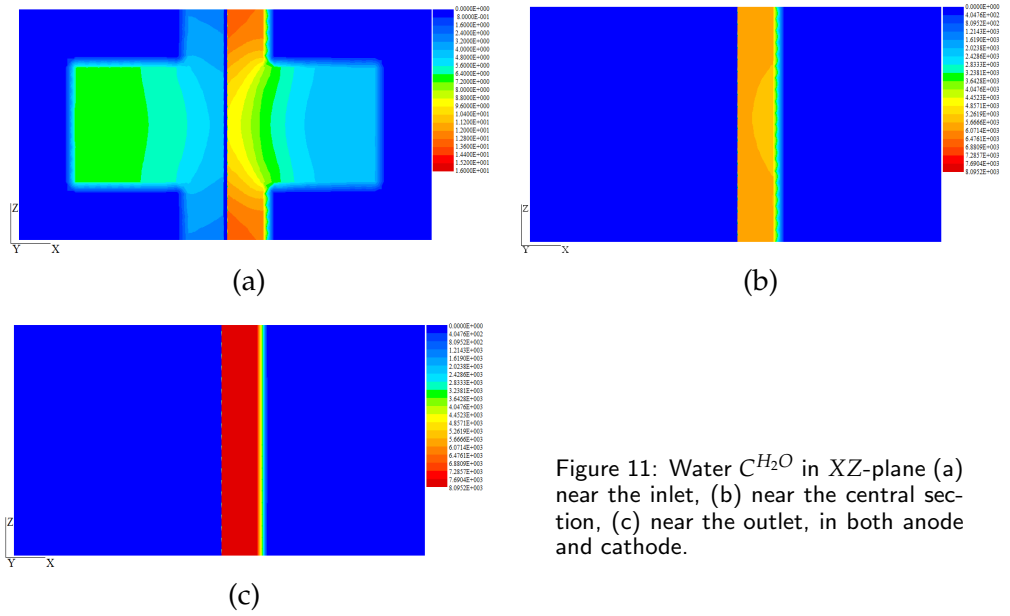


Figure 11: Water C^{H_2O} in XZ-plane (a) near the inlet, (b) near the central section, (c) near the outlet, in both anode and cathode.

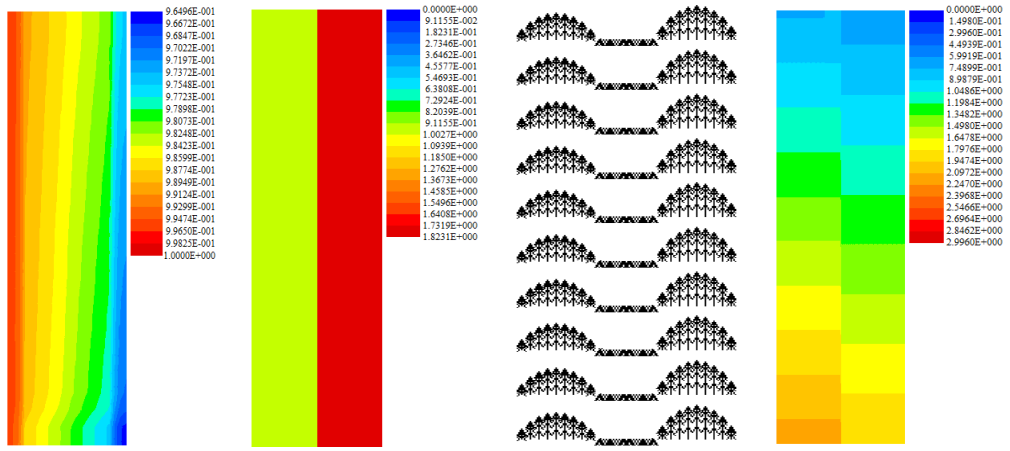


Figure 12: (a) Proton potential Φ_e in MEA; (b) Electron potential Φ_s in electrode; (c) Velocity field; (d) Pressure field, all in XY-plane.

Figs. 12(a) and (b) show the potential contours of proton and electron, where we can observe that the membrane electrode assembly (MEA) has a protonic potential gradient across the MEA from anode to cathode, which is due to the production of proton at anode catalyst layer as well as the consumption at cathode catalyst layer. Conversely, the electronic potential gradient is presented across the electrode from cathode to anode. The output average cell voltage is around 0.81 V obtained by subtracting the electronic potential at the surface of anode from that at the surface of cathode, corresponding to the

applied constant current density $I_{\text{ref}} = 0.2\text{A}/\text{cm}^2$. Fig. 12(d) shows that the channel has a pressure gradient along the channel due to the channel viscous flow, which will affect the gaseous flow in the diffusion media according to the Darcy's law and induce an along-channel component of the gas velocity. On the other hand, because the pure hydrogen is fed in the anode channel together with highly humid water vapor, while the highly dry air is fed in the cathode channel of PEMFC, $C_{\text{in}}^{O_2}$ is much smaller than $C_{\text{in}}^{H_2}$, thus the inlet velocity at cathode is bigger than that at anode due to (2.19), inducing a larger magnitude of velocity field in cathode gas channel than that in anode gas channel, as shown in Fig. 12(c).

The cell polarization curve is displayed in Fig. 13, qualitatively showing a good agreement with the experiments in [64].

In order to verify the correctness of our numerical solutions, we carry out the following numerical convergence study by doing simulations for PEMFC model (2.1)-(2.14) on a sequence of refined grids produced by a grid doubling, e.g., from $20\times4\times2$ to $160\times32\times16$ (four grids), and compare the obtained number of iteration, simulation time, convergence errors as well as mass balance errors on different mesh levels with increasing degree of freedoms (DOFs), as shown in Fig. 14 and Tables 3-5. To draw a fine graph for the tendencies of number of iteration and simulation time against increasing DOFs, we test more grids besides the above four ones.

In Fig. 14(a) we can see that the number of iteration drops from 70 more steps to 30 steps along with increasing degree of freedoms on a sequence of grids. Such decrease occurs very fast during the first four coarse grids, i.e., the number of iteration goes down to 32 steps or so on the grid with about 100,000 DOFs. Then it turns out to be stable at 30 iterations when grids are finer and finer. On the other hand, Fig. 14(b) shows that the simulation time grows slowly up to about 35 minutes on coarse grids in the range of 100,000 DOFs, and then increases very fast when DOFs are greater than 100,000. These computational phenomena indicate that we do not need much finer grid for PEMFC simulation in the sense of efficiency, i.e., less iteration step and less simulation time. A moderate grid with about 100,000 DOFs is good enough to produce an efficient numerical simulation for PEMFC model with about 30 steps in nonlinear iteration, 35 minutes in simulation time, as well as relatively accurate numerical results which are demonstrated at below.

Note that the finite element space S_h we use for the discretizations (3.8)-(3.14) is a piecewise linear polynomial space, it is well known that the following second order convergence error in L^2 norm holds for a pure linear finite element element on a quasi-uniform grid

$$\|u-u_h\|_0 = \mathcal{O}(h^2), \quad \text{for } u\in H^2(\Omega), \tag{6.1}$$

where $u_h \in S_h$ is the numerical solution of real solution u. However, we do not know the real solution u of PEMFC model (2.1)-(2.14). To investigate the convergence error for the obtained numerical solution u_h, we carry out the following error estimates based on the numerical solutions on a sequence of refined grids

$$\|u_{2^{j-1}h}-u_{2^jh}\|_0 = \|u_{2^{j-1}h}-u+u-u_{2^jh}\|_0 \le \|u-u_{2^{j-1}h}\|_0+\|u-u_{2^jh}\|_0,$$

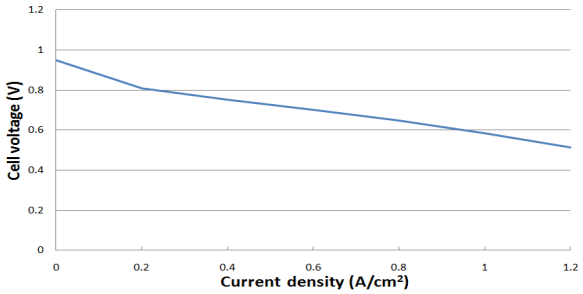
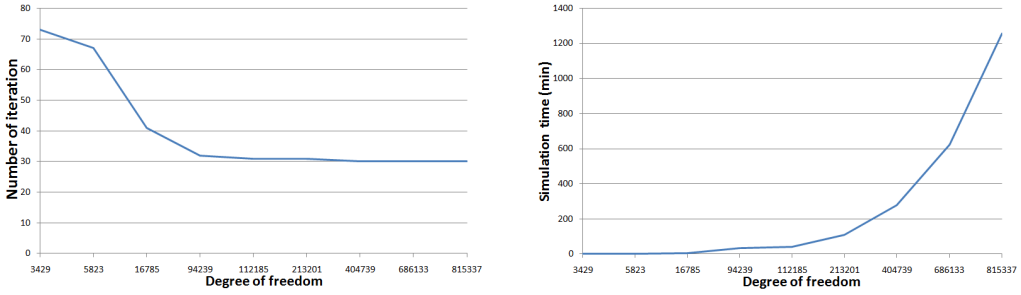


Figure 13: Polarization curve.



(a) (b)

Figure 14: (a) Number of iteration versus degree of freedom; (b) Simulation time versus degree of freedom.

leading to

$$\|u_{2^{j-1}h}-u_{2^jh}\|_0=\mathcal{O}\big((5\times 2^{2j-2})h^2\big), \tag{6.2}$$

where we apply (6.1) to two adjacent mesh levels with the mesh size $2^{j-1}h$ and 2^jh, respectively. Here $j=1,2,\cdots$, denoting the mesh level number, $j=1$ means the finest mesh with mesh size h, and the mesh size of j-th level mesh is $2^{j-1}h$. Thus, we have the following error indicator of seconder order convergence character

$$\frac{\|u_{2^jh}-u_{2^{j+1}h}\|_0}{\|u_{2^{j-1}h}-u_{2^jh}\|_0}\approx 4. \tag{6.3}$$

As for the combined finite element-finite volume method, the optimal error estimate (6.1) is replaced by the following sub-optimal error estimate [14]

$$\|u-u_h\|_0=\mathcal{O}(h^{1+\varepsilon}), \quad \text{for } u\in H^2(\Omega), \tag{6.4}$$

where $\varepsilon\in[0,1)$. Correspondingly, we substitute (6.3) by the following new error indicator

$$\frac{\|u_{2^jh}-u_{2^{j+1}h}\|_0}{\|u_{2^{j-1}h}-u_{2^jh}\|_0}\approx 2^{1+\varepsilon}. \tag{6.5}$$

Applying (6.3) to the numerical solutions of potentials of electron and proton, and (6.5) to the numerical solutions of concentrations of hydrogen, oxygen and water on

Table 3: Convergence errors of charge potentials on successive refined grids.

Resolution	DOFs	Electron		Proton	
		Conv	Ratio	Conv	Ratio
$20\times4\times2$	3429	1.47×10^{-5}	4.43	4.61×10^{-7}	4.52
$40\times8\times4$	16785	3.32×10^{-6}	4.17	1.02×10^{-7}	3.85
$80\times16\times8$	112185	7.95×10^{-7}	–	2.65×10^{-8}	–
$160\times32\times16$	815337	–	–	–	–

Table 4: Convergence errors of species concentrations on successive refined grids.

Resolution	Hydrogen		Oxygen		Water	
	Conv	Ratio	Conv	Ratio	Conv	Ratio
$20\times4\times2$	2.42×10^{-3}	2.64	1.05×10^{-3}	3.18	1.23	3.73
$40\times8\times4$	9.15×10^{-4}	3.39	3.30×10^{-4}	3.42	0.33	3.75
$80\times16\times8$	2.70×10^{-4}	–	9.66×10^{-5}	–	.088	–
$160\times32\times16$	–	–	–	–	–	–

the aforementioned four subsequent grids by a grid doubling, we eventually obtain an approximately optimal (second order) convergence character for charge potentials, as shown in Table 3 where the convergence error $\|u_{2^{j-1}h}-u_{2^jh}\|_0$ (labeled "Conv") is reduced by a factor of 4 or so (indicated by the values in columns "Ratio") for each doubling of resolution, and a sub-optimal convergence rate for species concentrations illustrated by Table 4 in which $\|u_{2^{j-1}h}-u_{2^jh}\|_0$ is reduced by a factor of $2^{1+\varepsilon}$ $(0.4<\varepsilon<0.91)$ for each doubling of resolution. In both tables, the columns labeled "Ratio" show the ratios of convergence errors between subsequent grid resolutions. Thus, we can conclude that the numerical solutions obtained from the numerical discretizations (3.8)-(3.14) are relatively accurate due to their consistent convergence characters with the well-established approximation theory.

Now we analyze the mass balance errors for each species to investigate the accuracy of their numerical discretizations. The mass balance error of species k can be calculated by

$$\frac{\left|F^k_{out}-F^k_{in}-S^k\right|}{F^k_{in}}=\frac{\left|\oint_{\partial\Omega_{\text{outlet}}}\left(C^ku_y\right)|_{\text{outlet}}d\tau-\oint_{\partial\Omega_{\text{inlet}}}\left(C^ku_y\right)|_{\text{inlet}}d\tau-\int_\Omega S^kdx\right|}{\oint_{\partial\Omega_{\text{inlet}}}\left(C^ku_y\right)|_{\text{inlet}}d\tau}, \tag{6.6}$$

where F^k_{out} and F^k_{in} represent the total flux of species k through outlets and inlets of gas channels in y-direction, respectively. The positive flux means it points in the positive direction of y-axis. S^k denotes the source of species k occurring in PEMFC, which may bear negative sign if the generation of species k is less than its consumption. (6.6) indicates that the net incremental mass equals the outcome due to the source term.

By plugging in (6.6) with the assigned species concentrations C^k_{inlet} and velocity $u_{y,\text{inlet}}$, and the computed C^k_{outlet}, $u_{y,\text{outlet}}$ and S^k based on a sequence of refined grids, and com-

Table 5: Mass balance errors of species on successive refined grids.

Resolution	Hydrogen		Oxygen		Water	
	MB	Ratio	MB	Ratio	MB	Ratio
$20\times4\times2$	0.2531	3.37	0.2488	3.97	0.8160	2.54
$40\times8\times4$	0.0752	5.19	0.0626	5.74	0.3215	3.99
$80\times16\times8$	0.0145	11.15	0.0109	3.76	0.0804	3.79
$160\times32\times16$	0.0013	–	0.0029	–	0.0212	–

puting the integrals in terms of a certain numerical quadrature, say, trapezoidal quadrature rule, we attain the errors of mass balance (labeled "MB") for each species on different mesh levels in Table 5. We compute the ratios of mass balance errors between subsequent grid resolutions and label it as "Ratio" in Table 5. The values in columns "Ratio" indicate that the convergence rates for mass balance errors vary between the order of 1.3 to 3.5. It is hard to say how the mass balance error is exactly related to mesh size. However, Table 5 does show us a convergent mass balance error for each species with the convergence rate of approximately second order, at least.

7 Conclusions and future work

In this paper, we introduce a novel automated program generator for finite element/finite volume method (FEPG) and its application to the simulation of a multiphysics proton exchange membrane fuel cell (PEMFC) model. It is the first time that we successfully apply Kirchhoff transformation to a 3D multiphase, multicomponent, multiphysics PEMFC model and achieve a fast convergent nonlinear iteration and accurate numerical solutions. In virtue of a high-level algorithm description language (script), we can efficiently implement our advanced numerical techniques for the simulation of a complex multiphysics PEMFC model on the FEPG platform. Numerical success for a 3D two-phase transport, multiphysics PEMFC model demonstrates the efficiency of both our numerical techniques and FEPG system. As for the future work, a more complicated PEMFC model with a view to nonisothermality and water back diffusion effect through membrane will be studied, and the corresponding efficient numerical method in terms of Kirchhoff transformation will be developed as well. On the other hand, the automatically generated finite element/finite volume source code can be further developed toward the application of adaptive and multilevel methods, even parallel computing by means of the parallel version of FEPG system (pFEPG), which will be illustrated in the future work.

Acknowledgments

Pengtao Sun is supported by NSF Grant DMS-0913757 and 111-Program for energy-saving and environment-friendly automobile (B08019) of China. Pengtao Sun was also

partially supported by State Key Laboratory of Scientific and Engineering Computing, Chinese Academy of Sciences during his visit in July, 2010. Su Zhou is supported by 863 Program (2008AA050403) and Shanghai Pujiang Talent Plan (08PJ1409) of China. Qiya Hu is supported by The Key Project of Natural Science Foundation of China G11031006, National Basic Research Program of China G2011309702 and Natural Science Foundation of China G10771178.

References

[1] `http://www.fegensoft.com/english/FEPG.htm`.
[2] The dune tutorial, `http://www.dune-project.org/`.
[3] The fenics-python tutorial, `http://docs.python.org/tutorial/`.
[4] H. W. Alt and S. Luckhaus, Quasilinear elliptic-parabolic differential equations, Math. Z., 183 (1983), 311–341.
[5] T. Arbogast, M. F. Wheeler and N. Zhang, A nonlinear mixed finite element method for a degenerate parabolic equation arising in flow in porous media, SIAM J. Numer. Anal., 33 (1996), 1669–1687.
[6] P. Asinari, M. C. Quaglia, M. R. von Spakovsky and B. V. Kasula, Direct numerical calculation of the kinematic tortuosity of reactive mixture flow in the anode layer of solid oxide fuel cells by the lattice boltzmann method, J. Power Sources, 170 (2007), 359–375.
[7] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. M. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine and H. V. der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, Philadalphia: Society for Industrial and Applied Mathematics, 1994.
[8] F. Brezzi and J. Pitkaranta, On the stabilization of finite element approximations of stokes problem, Efficient solutions of elliptic systems, Notes on Numerical Fluid Mechanics, Viewig 10 (1984), 11–19.
[9] H. S. Carslaw and J. C. Jaeger, Conduction of Heat in Solids, 2nd ed, Oxford University Press, Oxford, UK, 1959.
[10] Z. Chen, Formulations and numerical methods of the black oil model in porous media, SIAM J. Numer. Anal., 38 (2000), 489–514.
[11] M. A. Van Doormaal and J. G. Pharoah, Determination of permeability in fibrous porous media using the lattice boltzmann method with application to pem fuel cells, Int. J. Numer. Meth. Fluids, 59 (2009), 75–89.
[12] H. Elman and D. Silvester, Fast nonsymmetric iterations and preconditioning for navier-stokes equations, SIAM J. Sci. Comput., 17 (1996), 33–46.
[13] N. R. Eyres, D. R. Hartree, J. Ingham, R. Jackson, R. J. Sarjant and S. M. Wagstaff, The calculation of variable heat flow in solids, Philos. Trans. R. Soc. Lond., A240 (1946), 1–57.
[14] M. Feistauer, J. Felcman, M. Lukáčová-Medvid'Ová and G. Warnecke, Error estimates for a combined finite volume-finite element method for nonlinear convection-diffusion problems, SIAM J. Numer. Anal., 36 (1999), 1528–1548.
[15] S. Gottesfeld, Polymer Electrolyte Fuel Cells, Volume 5 of Advances in Electrochemical Science and Engineering, John Wiley & Sons, New York, 1997.
[16] T. J. R. Hughes, L. P. Franca and M. Balestra, A new finite element formulation for computational fluid dynamics: V. circumventing the babuska-brezzi condition: a stable petrov-galerkin formulation of the stokes problem accommodating equal-order interpolations, Comput. Meth. Appl. Mech. Eng., 59 (1986), 85–99.

[17] S. Ji, Y. Park, E. A. Sudicky and J. F. Sykes, A generalized transformation approach for simulating steady-state variably-saturated subsurface flow, Adv. Water Res., 31 (2008), 313–323.
[18] J. Kačur and R. V. Keer, Numerical approximation of a flow and transport system in unsaturated-saturated porous media, Chem. Eng. Sci., 58 (2003), 4805–4813.
[19] D. Kay, D. Loghin and A. Wathen, A preconditioner for the steady-state navier-stokes equations, SIAM J. Sci. Comput., 24 (2002), 237–256.
[20] A. Klawonn and G. Starke, Block triangular preconditioners for nonsymmetric saddle point problems: field-of-values analysis, Numer. Math., 81 (1999), 577–594.
[21] G. P. Liang, Finite Element Language, Science Press, Beijing, China, 2008.
[22] F. Q. Liu, G. Q. Lu and C. Y. Wang, Low crossover of methanol and water through thin membranes in direct methanol fuel cells, J. Electrochem. Soc., 153 (2006), A543–553.
[23] F. Q. Liu and C. Y. Wang, Optimization of cathode catalyst layer for direct methanol fuel cells: part II-computational modeling and design, Electrochim. Acta, 52 (2006), 1409–1416.
[24] F. Q. Liu and C. Y. Wang, Mixed potential in a direct methanol fuel cell-modeling and experiments, J. Electrochem. Soc., 154 (2007), B514–B522.
[25] W. Liu and C. Y. Wang, Modeling water transport in liquid feed direct methanol fuel cells, J. Power Sources, 164 (2007), 189–195.
[26] W. Liu and C. Y. Wang, Three-dimensional simulations of liquid feed direct methanol fuel cells, J. Electrochem. Soc., 154 (2007), B352–B361.
[27] D. Loghin and A. J. Wathen, Analysis of preconditioners for saddle-point problems, SIAM J. Sci. Comput., 25 (2004), 2029–2049.
[28] Z. Lu and D. Zhang, Analytical solutions to steady state unsaturated flow in layered, randomly heterogeneous soils via kirchhoff transformation, Adv. Water Res., 27 (2004), 775–784.
[29] M. J. Martinez and D. F. McTigue, A boundary integral method for steady flow in unsaturated porous media, Int. J. Numer. Anal. Meth. Geomech., 16 (1992), 581–601.
[30] D. B. McWhorster and D. K. Sunada, Exact integral solutions for two-phase flow, Water Resour. Res., 26 (1990), 399–413.
[31] M. N. Ozisik, Boundary Value Problems of Heat Conduction, Dover, New York, 2002.
[32] U. Pasaogullari, P. Mukherjee, C. Y. Wang and K. Chen, Anisotropic heat and water transport in a pefc cathode gas diffusion layer, J. Electrochem. Soc., 154 (2007), B823–B834.
[33] U. Pasaogullari and C. Y. Wang, Liquid water transport in gas diffusion layer of polymer electrolyte fuel cells, J. Electrochem. Soc., 151 (2004), A399–A406.
[34] U. Pasaogullari and C. Y. Wang, Two-phase modeling and flooding prediction of polymer electrolyte fuel cells, J. Electrochem. Soc., 152 (2005), A380–A390.
[35] S. Perng and H. Wu, Effects of internal flow modification on the cell performance enhancement of a pem fuel cell, J. Power Sources, 175 (2008), 806–816.
[36] S. Perng, H. Wu, T. Jue and K. Cheng, Numerical predictions of a pem fuel cell performance enhancement by a rectangular cylinder installed transversely in the flow channel, Appl. Energy, 86 (2009), 1541–1554.
[37] I. S. Pop, F. Radu and P. Knabner, Mixed finite elements for the richards' equation: linearization procedure, J. Comput. Appl. Math., 168 (2004), 365–373.
[38] S. A. Pourhashemi, O. J. Hao and R. C. Chawla, An experimental and theoretical study of the nonlinear heat conduction in dry porous media, Int. J. Energy Res., 23 (1999), 389–401.
[39] K. B. Prater, Polymer electrolyte fuel cells: a review of recent developments, J. Power Sources, 51 (1994), 129–144.
[40] F. A. Radu, I. S. Pop and S. Attinger, Analysis of an euler implicit-mixed finite element scheme for reactive solute transport in porous media, Numer. Methods Part. Differ. Eq., 26

(2010), 320–344.
[41] J. F. Rodrigues and J. M. Urbano, On a darcy-stefan problem arising in freezing and thawing of saturated porous media, Cont. Mech. Thermodyn., 11 (1999), 181–191.
[42] M. Rose, Numerical methods for flows through porous media I, Math. Comput., 40 (1983), 435–467.
[43] P. J. Ross and K. L. Bristow, Simulating water movement in layered and gradational soils using the kirchhoff transform, Soil Sci. Soc. Am. J., 54 (1990), 1519–1524.
[44] B. Schweizer, Regularization of outflow problems in unsaturated porous media with dry regions, J. Differential Equations, 237 (2007), 278–306.
[45] D. Silvester, H. Elman, D. Kay and A. Wathen, Efficient preconditioning of the linearized navier-stokes equations for incompressible flow, J. Comput. Appl. Math., 128 (2001), 261–279.
[46] P. C. Sui and N. Djilali, Analysis of coupled electron and mass transport in the gas diffusion layer of a pem fuel cell, J. Power Sources, 161 (2006), 294–300.
[47] P. T. Sun, C. Y. Wang and J. C. Xu, A combined finite element-upwind finite volume method for liquid-feed direct methanol fuel cell simulations, J. Fuel Cell Sci. Technol., 7 (2010).
[48] P. T. Sun and Y. Wang, A new formulation and an efficient numerical technique for a non-isothermal, anisotropic, two-phase transport model of pemfc, in Proceeding of Eighth International Fuel Cell Science, Engineering and Technology Conference, Brooklyn, New York, June 14-16, 2010, ASME.
[49] P. T. Sun, G. Xue, C. Y. Wang and J. C. Xu, A domain decomposition method for two-phase transport model in the cathode of a polymer electrolyte fuel cell, J. Comput. Phys., 228 (2009), 6016–6036.
[50] P. T. Sun, G. Xue, C. Y. Wang and J. C. Xu, Fast numerical simulation of two-phase transport model in the cathode of a polymer electrolyte fuel cell, Commun. Comput. Phys., 6 (2009), 49–71.
[51] P. T. Sun, G. Xue, C. Y. Wang and J. C. Xu, New numerical techniques for a liquid feed 3d full direct methanol fuel cell model, SIAM Appl. Math., 70 (2009), 600–620.
[52] D. M. Tartakovsky, Prediction of steady-state flow of real gases in randomly heterogeneous porous media, Physica D, 133 (1999), 463–468.
[53] D. M. Tartakovsky, A. Guadagnini and M. Riva, Stochastic averaging of nonlinear flows in heterogeneous porous media, J. Fluid Mech., 492 (2003), 47–62.
[54] D. M. Tartakovsky, S. P. Neuman and Z. Lu, Conditional stochastic averaging of steady state unsaturated flow by means of kirchhoff transformation, Water Resour. Res.,, 35 (1999), 731–745.
[55] T. E. Tezduyar, Stabilized finite element formulations for incompressible flow computations, Adv. Appl. Mech., 28 (1992), 1–44.
[56] R. Tirnovan, S. Giurgea, A. Miraoui and M. Cirrincione, Proton exchange membrane fuel cell modelling based on a mixed moving least squares technique, Int. J. Hydrogen Energy, 33 (2008), 6232–6238.
[57] R. Tirnovan, S. Giurgea, A. Miraoui and M. Cirrincione, Surrogate model for proton exchange membrane fuel cell (pemfc), J. Power Sources, 175 (2008), 773–778.
[58] P. Vadasz, Analytical solution to nonlinear thermal diffusion: Kirchhoff versus cole-hopf transformations, J. Heat Trans., 132 (2010), 121302-1–121302-6.
[59] S. Wan, B. Wu and N. Chen, Application of program generation technology in solving heat and flow problems, J. Therm. Sci., 16 (2007), 170–175.
[60] C. Y. Wang, Fundamental models for fuel cell engineering, Chem. Rev., 104 (2004), 4727–

4766.
[61] C. Y. Wang and P. Cheng, Multiphase flow and heat transfer in porous media, Adv. Heat Trans., 30 (1997), 93–196.
[62] Y. Wang, Modeling of two-phase transport in the diffusion media of polymer electrolyte fuel cells, J. Power Sources, 185 (2008), 261–271.
[63] Y. Wang and C. Y. Wang, A nonisothermal two-phase model for polymer electrolyte fuel cells, J. Electrochem. Soc., 153 (2006), A1193–A1200.
[64] Y. Wang, C. Y. Wang and K. S. Chen, Elucidating differences between carbon paper and carbon cloth in polymer electrolyte fuel cells, Electr. Acta, 52 (2007), 3965–3975.
[65] Z. H. Wang, C. Y. Wang and K. S. Chen, Two-phase flow and transport in the air cathode of proton exchange membrane fuel cells, J. Power Sources, 94 (2001), 40–50.
[66] B. Wu, H. Qian and S. Wan, Promotion of frontier science research by aid of automatic program generation technology, in Computational Methods in Engineering & Science, Proceedings of "Enhancement and Promotion of Computational Methods in Engineering and Science X", Sanya, China, Aug 21-23, 2006, Tsinghua University Press & Springer-Verlag.
[67] X. Zhua, P. C. Sui and N. Djilali, Dynamic behaviour of liquid water emerging from a gdl pore into a pemfc gas flow channel, J. Power Sources, 172 (2007), 287–295.