AN OPEN-SOURCE TOOLBOX FOR SINGLE-PHASE FLOW SIMULATION IN PEM FUEL CELLS

Jean-Paul Kone[1]

1. INTRODUCTION

An open-source code toolbox capable of predicting the distribution of the major physical quantities which are transported within a PEM fuel cell has been created using OpenFOAM. The toolbox can be used to rapidly gain important insights into the cell working processes which are essential for design optimization. It consists of a main program, relevant library classes, and a constructed simulation case for a co-flow galvanostatic run.

2. SOURCE CODE

2.1 LIBRARY SOURCE CODE

The libraries of the pemfcModels-4.0 toolbox are in the `lib/` directory. Table 1 contains a complete list of the libraries (the names are reasonably descriptive).

Table 1 Libraries

Library name	Description
`diffusivityModels`	Diffusivity models used for laminar incompressible flow with multi-component mixtures and/or porous media
`myContinuityErrs`	Calculates and prints the continuity errors
`MyMeshWave`	Wave propagation of information through the grid
`myPatchToPatchInterpolation`	Interpolation class dealing with transfer of data between two primitivePatches
`myPorosityModel`	Porosity models with input/output functionality
`pemfcSpecie`	A set of electrochemical properties for a PEM fuel cell species
`polyToddYoung`	Todd-Young polynomials for gases thermodynamics properties calculation
`regionProperties`	Holds region information for coupled region simulations
`smearPatchToMesh`	Smears data from patch to mesh, giving each cell the value of the nearest patch face

[1] International Doctoral Innovation Centre IDIC, University of Nottingham, Ningbo 315100, China.

2.2 SOLVER SOURCE CODE

The program starts in the main function, coded in file `pemfcSinglephaseNonIsothermalSolver.c`, through the model execution. The model then begins by including the required OpenFOAM source code files to check the case path, and to read the `system/controlDict` file and instantiate the `Time` object `runTime`. This is followed by the generation of meshes, the reading of properties, and the creation of fields for the global cell mesh and the region meshes. The different components of the solver code are listed in Table 2, along with a brief description of each component.

Table 2 Description of the solver source code

Folder/file name	Description
`pemfcSinglephaseNon-IsothermalSolver.C`	Solver that can simulate the single-phase non-isothermal operation of PEM fuel cell
`constants`	Holds the physical constant values (e.g. universal gas constant, Faraday's constant, etc.)
`createFields`	Creates field variables (e.g. temperature, velocity, pressure, etc.) for the entire cell, and for the individual cell components
`createMesh`	Creates mesh for the entire cell, and for the individual cell components
`createSpecies`	Creates the individual gas species in the fuel (e.g. hydrogen and water vapour) on the anode side, and in air (e.g. oxygen, nitrogen and water vapour) on the cathode side. Calculates the mole fraction fields from the mass fractions fields of the gas species
`diffusivity`	Creates the gas diffusivity models of both fuel and air, and calculates the diffusivity values for the individual gas species

Table 2 (Continued)

Folder/file name	Description
`electrochemistry`	Calculates electrochemistry (e.g. cell current density, cell voltage, Nernst potential, overpotentials, electrochemical heating, etc.)
`energyTransport`	Solves the energy equation for the global mesh
`mapping`	Map regional fields to global mesh to solve the energy equation. Map global temperature to fluid region meshes to calculate local fluid density
`momentumTransport`	Solves pressure and momentum using the PISO iteration for both fuel and air. Following the solution, the Reynolds numbers are calculated (this is informative only)
`multiSpeciesTransport`	Solves the species transport equation for each specie other than the background (inert) specie. The mass fraction of the inert specie is computed by subtracting the sum of the mass fractions of all the other species from unity
`patch`	Sets global variables for the IDs of many patches that are frequently referenced. Creates patch to patch interpolation for interpolating mole fraction fields of fuel on the air since electrochemistry is assumed to occur on air/electrolyte interface
`physicalProperties`	Computes physical properties such as thermal conductivity, dynamic viscosity and density for both fuel and air
`readParameters`	Reads the activation parameters
`readProperties`	Reads properties of the entire cell and individual cell components, as well as properties of the reactants gases and the reaction
`tools`	Holds various tools for function integration, Ridder's method of root finding, etc.

2.3 COMPILING THE SOURCE CODE

Assuming OpenFOAM version 4.0 is installed, with the environmental variables set. To compile the library and application source code, go to:

```
pemfcModels-4.0/singlePhaseNonIsothermal
```

directories and run the (`Allwmake`) script. To do this, type (`./Allwmake`) at the command terminal. This should generate the shared object library `libSinglephasePEMFC.so` in the `$FOAM_USER_LIBBIN` directory and application executable `pemfcSinglephaseNonIsothermalSolver` in the `$FOAM_USER_APPBIN` directory.

A `lnInclude/` directory, containing links to all the `lib` class files, will appear in the `lib/` directory.

3. MODEL RUN

3.1 RUNNING A SIMULATION

Assuming OpenFOAM version 4.0 is installed, with the environmental variables set and the `pemfcSinglephaseNonIsothermalSolver` application already compiled, the command (`make run`) will run the executable from the terminal using the constructed case data. The model can also be run by typing the executable name and the output that is normally directed to Standard Out can be redirected to a file:

```
pemfcSinglephaseNonIsothermalSolver | tee log.run
```

After the model has run to completion, VTK files for visualization, e.g. with ParaView, can be created using the `Makefile` file by typing (`make view`) at the command line to generate VTK files for the last output step and (`make viewAll`) to get VTK files for all output directories. Otherwise, the (`paraFoam`) utility supplied with OpenFOAM can be invoked directly for visualization of results with ParaView.

3.2 SIMULATION INPUTS

Runtime inputs to the model are supplied in dictionaries in the case directory. Amongst these are the mesh files and the mesh mapping files generated during the mesh generation. Table 3 and 4 contain the complete list of the fields variables and the parameters that must be specified. The specifications supplied for the constructed case study can be viewed in the case files.

Numerical schemes are specified at runtime by `fvSchemes` files in the `system` directories. The `fvSchemes` dictionary contains sub-dictionaries which must be defined for the code to run. The `fvSchemes` used by the model are listed in Table 5, along with an indication of the regions in which they are applicable.

The solvers and other algorithmic controls and tolerances are supplied by the `fvSolution` dictionary files in the system directories as given in Table 6. The table shows three sub-dictionaries in the `fvSolution` files: `solvers`, `PISO`, and `relaxationFactors`. In the solvers sub-dictionary, the settings for the linear solvers chosen to solve the discretized finite volume equations for the various fields are specified. The `relaxationFactors` sub-dictionary contains under-relaxation factors to improve stability. The `PISO` sub-dictionary controls the PISO algorithm for the simultaneous solution of pressure and momentum. Table 6 also shows in which regions the settings are applied. Note that the `fvSolution` file must exist in the system directory even though it may not need any sub-dictionaries.

Table 3 Simulation input parameters and properties

file constant/cellProperties	
Parameter/property	*Comments*
`voltage`	initial value of voltage
`Ibar0`	prescribed mean current density
`fuelCellType`	fuel cell type
`Rhat`	voltage correction coefficient
`Tinit`	initial temperature of fuel and air
`kappaCl`	electronic conductivity of catalyst layer
`kappaGdl`	electronic conductivity of gas diffusion layer
`kappaBp`	electronic conductivity of bipolar plate
`rContact`	contact resistance between the gas diffusion layer and the bipolar plate
`tCl`	thickness of catalyst layer
`tGdl`	thickness of gas diffusion layer
`tBp`	thickness of bipolar plate
`epsilonGdl`	porosity of gas diffusion layer
`epsilonCl`	porosity of catalyst layer
`etaConConstant`	concentration overpotential constant
`fuelInletPatch`	fuel mesh patch name for the inlet patch
`fuelOutletPatch`	fuel mesh patch name for the outlet patch
`anodePatch`	fuel mesh patch name for the fuel/electrolyte interface
`fuelAbpPatch`	fuel mesh patch name for the fuel/anode bipolar plate interface
`airInletPatch`	air mesh patch name for the inlet patch
`airOutletPatch`	air mesh patch name for the outlet patch
`cathodePatch`	air mesh patch name for the air/electrolyte interface
`airCbpPatch`	air mesh patch name for the air/cathode bipolar plate interface
`electrolyteAnodePatch`	electrolyte mesh patch name for the electrolyte/fuel interface
`electrolyteCathodePatch`	electrolyte mesh patch name for the electrolyte/air interface
`abpFuelPatch`	anode bipolar plate mesh patch name for the anode bipolar plate /fuel interface
`cbpAirPatch`	cathode bipolar plate mesh patch name for the cathode bipolar plate /air interface
file constant/rxnProperties	
Parameter/property	*Comments*
`rxnSpecies`	list of species name and stoichiometric coefficient pairs
file constant/abp/abpProperties	
Parameter/property	*Comments*
`rho`	density of anode bipolar plate
`Cp`	heat capacity of anode bipolar plate
`k`	thermal conductivity of anode bipolar plate

Table 3 (Continued)

`file constant/air/airProperties`	
`Parameter/property`	*Comments*
`rho`	initial density of air mixture
`mu`	dynamic viscosity of air mixture
`Cp`	initial isobaric heat capacity of air mixture
`k`	thermal conductivity of air mixture
`dHyd`	hydraulic diameter for Reynolds number
`diffusivity`	air diffusivity model sub-dictionary
`file constant/air/pemfcSpeciesProperties`	
Parameter/property	*Comments*
`speciesList`	list of gas species in the air mixture
`O2`	properties of oxygen
`N2`	properties of nitrogen
`H2O`	properties of water vapour
`CpCoeffs`	heat capacity coefficients sub-dictionary
`muCoeffs`	dynamic viscosity coefficients sub-dictionary
`kCoeffs`	thermal conductivity coefficients sub-dictionary
`file constant/air/porosityProperties`	
Parameter/property	*Comments*
`cgdl`	cathode gas diffusion layer
`ccl`	cathode catalyst layer
`porosity`	porosity value
`Cp`	heat capacity
`k`	thermal conductivity
`myDarcyForchheimerCoeffs`	Darcy-Forchheimer sub-dictionary
`diffusivity`	diffusivity model sub-dictionary
`file constant/cbp/cbpProperties`	
same as for `abpProperties`, but for `cbpProperties`	
`file constant/electrolyte/activationParameters`	
Parameter/property	*Comments*
`i0Reference`	reference exchange current density
`ECathode`	activation energy
`alphaCathode`	transfer coefficient
`file constant/electrolyte/electrolyteProperties`	
Parameter/property	*Comments*
`rho`	density of electrolyte
`Cp`	heat capacity of electrolyte
`k`	thermal conductivity of electrolyte
`sigma`	initial ionic conductivity of electrolyte
`file constant/fuel/fuelProperties`	
same as for `airProperties`, but for `fuelProperties`	
`file constant/fuel/pemfcSpeciesProperties`	
same as for air species, but for fuel species	
`file constant/fuel/porosityProperties`	
same as for air porous zones, but for fuel porous zones	

Table 4 Simulation input initial fields

File	**Field variable**	**Comments**
`0/k`	cell thermal conductivity	May be changed to suit operating conditions
`0/T`	cell temperature	Inlet values = 1e-15 prevents outward diffusion at inlets
`0/air/diffO2air`	diffusivity of Oxygen in air mixture	Inlet value = 1e-15 prevents outward diffusion at inlet Not required if outward diffusion at inlet is not an issue Never required for background specie
`0/air/p`	air pressure	internalField and outlet boundaries at atmospheric pressure; other patches zeroGradient or equivalent
`0/air/rho`	air density	Optional
`0/air/T`	air temperature	Optional
`0/air/U`	air velocity	internalField 0 (or initialized to inlet value); inlet specified; outlet zeroGradient; cathodePatch type must allow code to set value (e.g. fixedValue)
`0/air/YSpair`	mass fraction of specie Sp	internalField initialized to inlet value cathodePatch must be type fixedGradient. Require one such file for each air specie, e.g., YH2Oair, YN2air, YO2air
`0/fuel/diffH2fuel`	diffusivity of hydrogen in fuel mixture	same as for 0/air/diffO2air
`0/fuel/p`	fuel pressure	same as for 0/air/p
`0/fuel/rho`	fuel density	same as for 0/air/rho
`0/fuel/T`	fuel temperature	same as for 0/air/T
`0/fuel/U`	fuel velocity	same as for 0/air/U
`0/fuel/YSpfuel`	mass fraction of species Sp	same as for 0/air/YSpair e.g., YH2fuel, YH2Ofuel

Table 5 Simulation input `fvScheme` settings

Operator	Scheme	Region(s)
ddtSchemes		
default	`steadyState;`	all
gradSchemes		
default	`Gauss linear;`	all
grad(p)	`Gauss linear;`	air, fuel
divSchemes		
default	`none;`	all
div(rhoCpPhi,T)	`Gauss upwind;`	cell
div(phi,U)	`Gauss GammaV 0.2;`	air, fuel
div(phi,y)	`Gauss upwind;`	air, fuel
laplacianSchemes		
default	`none;`	all
laplacian(k,T)	`Gauss harmonic corrected;`	cell
laplacian(mu,U)	`Gauss harmonic corrected;`	air, fuel
laplacian((rho|A(U)), p)	`Gauss linear corrected;`	air, fuel
laplacian(gamma,y)	`Gauss harmonic corrected;`	air, fuel
laplacian(diff,y)	`Gauss harmonic corrected;`	air, fuel
interpolationSchemes		
default	`linear;`	all
interpolate(T)	`harmonic;`	cell, air, fuel
interpolate(k)	`harmonic;`	cell, air, fuel
interpolate(rho)	`harmonic;`	air, fuel
snGradSchemes		
default	`corrected;`	all
fluxRequired		
default	`no;`	all
T		cell
p		air, fuel

Table 6 Simulation input `fvSolution` settings

Solver dictionary			
Field	*Solver*	*Parameters*	*Region(s)*
T	PBiCG	`preconditioner DILU;` `tolerance 1e-18;` `relTol 0.0;` `minIter 1;` `maxIter 5000;`	cell
p	PCG	`preconditioner DIC;` `tolerance 1e-9;` `relTol 0.0;` `minIter 1;` `maxIter 700;`	air, fuel
U	PBiCG	`preconditioner DILU;` `tolerance 1e-9;` `relTol 0.0;` `minIter 1;` `maxIter 700;`	air, fuel
Yi	PBiCG	`preconditioner DILU;` `tolerance 1e-9;` `relTol 0.0;` `minIter 1;` `maxIter 1000;`	
PISO dictionary			
Parameter		*Value*	*Region(s)*
`nIteration`		`0;`	air, fuel
`nCorrectors`		`2;`	
`nNonOrthogonalCorrectors`		`0;`	
`pRefCell`		`0;`	
`pRefValue`		`0;`	
`relaxationFactors` dictionary			
Field		*Value*	*Region(s)*
p		`0.3;`	air, fuel
U		`0.7;`	
yi		`0.5`	

3.3 SIMULATION OUTPUTS

Not only the model writes selected fields to time directories in the case directory, but it also writes to Standard Out as it proceeds.

The model produces time directories in the case directory, in accordance with the settings in the control dictionary (`system/controlDict`). For a steady state model like `pemfcSinglephaseNonIsothermalSolver`, these directory time name (e.g. `60/`, `120/`, etc.) represent the iteration count rather than time. Field `IOobjects` created with the `AUTO_WRITE` attribute as given in Table 7, will be written to these time directories.

Table 7 Simulation outputs at time > 0

<case>/	**/abp/**	**/air/**	**/cbp/**	**/electrolyte/**	**/fuel/**	**field variable**
		cp			cp	heat capacity
		diffSpair			diffSpfuel	diffusivity of species Sp
*k		k			k	thermal conductivity
		p			p	pressure
		phi			phi	velocity flux
		rho			rho	density
*T	T	T	T	T	T	temperature
		*U			*U	velocity
		XSpair			XSpfuel	mole fraction of species Sp
		*YSpair			*YSpfuel	mass fraction of species Sp
Tsource						energy source
				electrochemicalHeating		electrochemical heating
				etaActC		activation overpotential
				etaConC		concentration overpotential
				etaOhmic		ohmic overpotential
				I		current density
				i0C		exchange current density
				iLC		limiting current density
				NernstPot		Nernst potential
				sigmaMem		ionic conductivity
				V		voltage

*are MUST_READ and thus required at time 0