A MODEL FOR 1D MULTICOMPONENT GAS TRANSPORT IN THE GAS DIFFUSION LAYER OF THE PROTON EXCHANGE MEMBRANE FUEL CELL

GREG LUKEMAN

Abstract. One of the most promising emerging green energy technologies is the proton exchange membrane fuel cell, an electrochemical energy conversion device that converts hydrogen and oxygen into electricity and heat. In this device, a mixture of gases, including hydrogen, is introduced to a catalyst via a semi-permeable gas diffusion layer (GDL). The efficiency of the fuel cell is in part governed by the composition of the GDL and the composition of the gas mixture. In this report, we motivate the problem and discuss a one-dimensional mathematical model of the multicomponent gas transport across the GDL suggested by Stockie in [5], consisting of coupled parabolic partial differential equations and some separated boundary conditions. We solve these equations with `BACOL`, a high-order adaptive method of lines package based on B-spline collocation. We compare our results with those obtained by Stockie [5] and discuss future work.

Date: Friday, December 13, 2002.

1. Introduction

Non-polluting energy production has become an important area of research in recent years and is poised to become even more urgent with the passage of international environmental agreements. One of the most promising areas of research is the Proton Exchange Membrane fuel cell. The PEM fuel cell is an electrochemical device that converts hydrogen and oxygen into electricity and heat with water as the by-product. These devices are compact in size and produce a reasonable amount of energy at relatively low temperatures (80° C) and thus have a wide range of applications.

In the following section, we take a brief look at the chemistry and physics of the fuel cell. We will then introduce a model for an integral component of the fuel cell; the gas diffusion layer (GDL). In subsequent sections we discuss the implementation of this model with `BACOL`, a high-order adaptive method of lines package based on B-spline collocation, compare the results with a fixed mesh implementation in Matlab by Stockie, and look forward to some future directions.

2. The PEM Fuel Cell

The basic elements of a PEM fuel cell (Figure (1)) are the anode, the cathode, the proton exchange membrane, and the catalyst. The

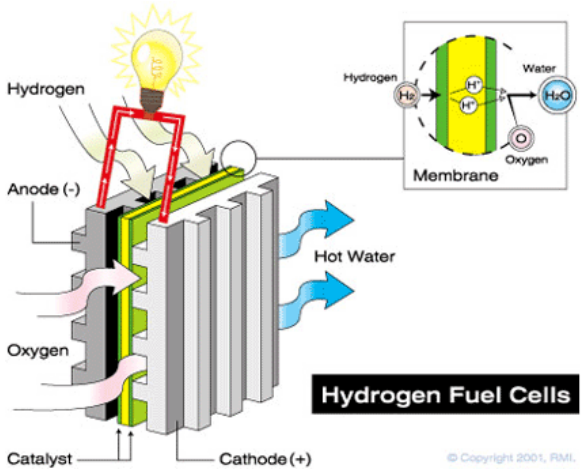

FIGURE 1. Schematic of the PEM fuel cell

anode is where the hydrogen gas is introduced to the catalyst through a gas diffusion layer. The GDL is a highly porous material (such as carbon fiber paper) which evenly distributes the reactant gas to the catalyst. A *catalyst* is a substance that changes the rate of a chemical reaction by providing an alternate pathway to the final product. In the case of PEM fuel cells the catalyst is usually platinum powder. When the hydrogen gas (H_2) meets the catalyst, it splits into two H^+ ions and two electrons (e^-). The electrons are conducted by the anode to an external circuit where they travel to the cathode. It is this flow of electrons that produces the energy. The hydrogen ions travel through a *proton exchange membrane*, a special material that will only permit the passage of positively charged ions, to the cathode. At the *cathode* side, oxygen (O_2) from the atmosphere flows through a GDL towards its

own catalyst where it splits into two negatively charged oxygen atoms. The negative charge of these atoms pulls the hydrogen ions through the PEM. These ions, after traversing the PEM, combine with the oxygen atoms and the electrons that have travelled the external circuit to form water (H_2O).

2.1. **Chemistry.** The complete chemical reaction is:

$$\begin{aligned} \text{Anode side :} \quad & 2H_2 \Rightarrow 4H^+ + 4e^- \\ \text{Cathode side :} \quad & O_2 + 4H^+ + 4e^- \Rightarrow 2H_2O \\ \text{Net reaction :} \quad & 2H_2 + O_2 \Rightarrow 2H_2O. \end{aligned}$$

This reaction in a single cell only produces about 0.7 volts, so to achieve a usable voltage many cells are combined to form a fuel cell stack. This entire stack is often referred to as a fuel cell.

One challenge with this technology lies in its use of hydrogen gas. Hydrogen is not a readily available fuel and is dangerous to store in pure form. It is clear that we need an inexpensive and safe way to provide the hydrogen to the anode. This is done through a device called a reformer. A reformer extracts hydrogen from widely available fuels such as alcohol or methanol. Unfortunately, the reformer also

produces waste gases such as CO_2[1] during this conversion, and so the hydrogen gas that reaches the GDL is actually in a mixture of other gases.

2.2. **Gas Diffusion Layer.** The constituents and relative quantities of these other gases in the mixture, along with the composition of the GDL can largely govern the efficiency of a fuel cell. Clearly, it would be useful to have an accurate model of what is happening at this crucial stage of a fuel cell's operation: the transport of the gas mixture through the GDL. There has been much research done in this area, often treating the gas flow in 2-dimensions ([4] and [8] for example) and concentrating on various complexities, such as temperature dependence and condensation. These models look to the behavior of simple 1-dimensional models for calibration and verification. The remainder

[1]One would be justified in raising the following question at this point. Since pollution control is one of our objectives, how are fuel cells any cleaner than the internal combustion engine if they end up producing CO_2 anyway? The answer lies in the efficiency of the energy production. The production of mechanical work in a fuel cell powered car is about 32% efficient (and improving), while a gasoline powered car is about 20% efficient. The overall efficiency is further improved when we consider that a fuel cell powered car can be much lighter, since much of the weight of a common automobile is the internal combustion engine itself.

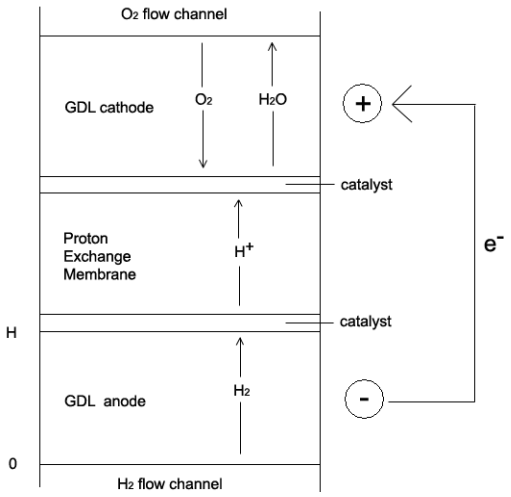

FIGURE 2. Transport through the GDL

of this report will deal with one such simple 1-d model, suggested by Stockie in [5].

3. GDL MODEL

Consider the cross-section of the PEM fuel cell shown in Figure (2). Let our spatial co-ordinate be $y \in [0, H]$. Although in Figure (2) we have $[0, H]$ labelled on the anode side, the model is equally applicable to the cathode side, with the constants changed to appropriate values.

The governing equations are the porous medium equation,

$$\frac{\partial C}{\partial t} - \frac{\partial}{\partial y}\left(\Gamma C \frac{\partial C}{\partial y}\right) = 0, \tag{3.1}$$

and the convection-diffusion equation,

$$\frac{\partial R}{\partial t} - \frac{\partial}{\partial y}\left[\Gamma R \frac{\partial C}{\partial y} + DC\frac{\partial}{\partial y}\left(\frac{R}{C}\right)\right] = 0, \tag{3.2}$$

where the variables are defined in the following table.

C	mixture concentration
R	reactant concentration
D	diffusivity
Γ	$\frac{K\mathcal{R}\mathcal{T}}{\epsilon\mu}$
K	permeability
$\mathcal{R}$	gas constant
$\mathcal{T}$	temperature
ϵ	porosity
μ	viscosity

The boundary conditions at $y = 0$ are

$$C = \overline{C} \quad \text{and} \quad \tfrac{\partial}{\partial y}\left(\tfrac{R}{C}\right) = -\tfrac{r_0}{D\overline{C}}(\overline{R} - R) \tag{3.3}$$

and the boundary conditions at $y = H$ are

$$\tfrac{\partial}{\partial y}\left(\tfrac{R}{C}\right) = -\tfrac{r_H R}{DC}(\overline{R} - R) \quad \text{and} \quad \tfrac{\partial C}{\partial y} = -\tfrac{\nu r_H R}{\Gamma(C - \nu R)} \tag{3.4}$$

where $\overline{C}$ and $\overline{R}$ are the constant mixture and reactant concentrations within the flow channel at $y = 0$, $\nu = -1$ at for the cathode and 1

for the anode, r_0 is the bottom transfer rate and r_H is the top transfer rate. Stockie suggests default values for all these parameters in [5] for both the anode and cathode cases.

3.1. **Fixed Mesh Discretization.** Stockie suggests ([5],[4]) solving the nonlinear system of PDEs (3.1), (3.2) by first discretizing in space using the method-of-lines. The domain $[0, H]$ is divided into n cells of size $h = H/n$. Concentrations are taken at the cell centers $y_{i+1/2} = (i+1/2)/h$ where $i = 0, 1, \ldots, n-1$. The following system is obtained:

$$\begin{aligned}
\frac{dR_{i+1/2}}{dt} &= \frac{\Gamma}{h^2}[R_{i+1}(C_{i+3/2} - C_{i+1/2}) \\
&- R_i(C_{i+1/2} - C_{i-1/2})] \\
&+ \frac{d}{h^2}\left[C_{i+1}\left(\frac{R_{i+3/2}}{C_{i+3/2}} - \frac{R_{i+1/2}}{C_{i+1/2}}\right)\right. \\
&- \left. C_i\left(\frac{R_{i+1/2}}{C_{i+1/2}} - \frac{R_{i-1/2}}{C_{i-1/2}}\right)\right] \\
\frac{dC_{i+1/2}}{dt} &= \frac{\Gamma}{h^2}[C_{i+1}(C_{i+3/2} - C_{i+1/2}) \\
&- C_i(C_{i+1/2} - C_{i-1/2})].
\end{aligned}$$

The boundary conditions are approximated through the use of ghost points $y_{-1/2}$ and $y_{n+1/2}$ which lie at one half point outside the boundaries. This allows the $y = 0$ boundary condition to be second order accurate in space, as are the discretized equations. The $y = 0$ boundary

conditions become

$$\frac{C_{1/2}+C_{-1/2}}{2}=\overline{C}\Rightarrow C_{-1/2}=2\overline{C}-C_{1/2} \tag{3.5}$$

$$\frac{1}{h}\left(\frac{R_{1/2}}{C_{1/2}}-\frac{R_{-1/2}}{C_{-1/2}}\right)=-\frac{r_0}{D\overline{C}}\left(\overline{R}-\frac{1}{2}(R_{-1/2}+R_{1/2})\right)\Rightarrow \tag{3.6}$$

$$R_{-1/2}=\left(\frac{1}{C_{-1/2}}+\frac{hr_0}{2D\overline{C}}\right)^{-1}\left[\frac{R_{1/2}}{C_{1/2}}+\frac{hr_0}{D\overline{C}}\left(\overline{R}-\frac{R_{1/2}}{2}\right)\right].$$

Stockie found the $y=H$ boundary to be more of an issue as a nonlinear system needs to be solved in order to calculate the ghost points by a 2nd order method. Instead, without encountering any instability, he employs a 1st order upwind approximation at this boundary:

$$C_{n+1/2}=C_{n-1/2}-\frac{\nu h r_H R_{n-1/2}}{\Gamma(C_{n-1/2}-\nu R_{n-1/2})}, \tag{3.7}$$

$$\frac{R_{n+1/2}}{C_{n+1/2}}=\frac{R_{n-1/2}}{C_{n-1/2}}-\frac{hr_H R_{n-1/2}}{DC_{n-1/2}}. \tag{3.8}$$

3.2. **Stiffness.** Stockie notes in [4] that the solution of the PDE system (3.1), (3.2) has both fast and slow time scales, suggesting a very stiff problem. Initially, the solution is dominated by rapidly varying transients, necessitating a small time step while at later times the solution slowly approaches steady state and larger steps can be taken. The fixed spatial mesh approach just described is implemented by Stockie in a Matlab code resulting from [4]. In this code, the discretized ODE

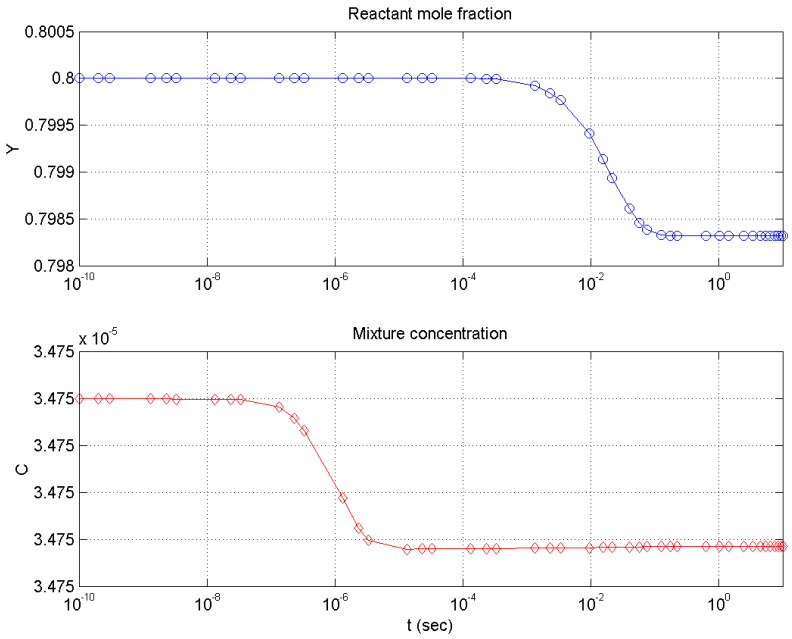

FIGURE 3. Matlab solution with fixed spatial mesh

system is solved by Matlab's `ode15s`, a variable-order (1 to 5) NDF (numerical differentiation formula) package. This implicit solver uses quasi-constant step sizes and is well suited to stiff problems [1]. The results from the Matlab implementation are shown in Figure (3) (R is shown as a fraction of the mixture). The fast initial time scale is evidenced by the need for a log scale in time to resolve the initial features.

4. BACOL

As an alternative to Stockie's fixed mesh approach, we solve the system (3.1), (3.2) with an adaptive mesh package. `BACOL`, authored by Rong Wang [7], is a high-order adaptive method-of-lines package for solving 1-dimensional parabolic partial differential equations of the form

$$u_t(x,t) = F(t,x,u(x,t),u_x(x,t),u_{xx}(x,t)), \qquad x_a \le x \le x_b, \quad t \ge t_0,$$

with initial conditions given by

$$u(x,t_0) = u_0(x), \qquad x_a \le x \le x_b,$$

and separated boundary conditions of the form

$$\begin{aligned} b_L(t,u(x_a,t),u_x(x_a,t)) &= 0, \quad t \ge t_0, \\ b_R(t,u(x_b,t),u_x(x_b,t)) &= 0, \quad t \ge t_0. \end{aligned}$$

The u can be vectors (for systems of PDEs). `BACOL` requires that the u_{xx} term not vanish in the right-hand side of the PDEs. Our system (3.1),(3.2) clearly conforms to these conditions.

`BACOL` uses collocation with a B-spline basis for the spatial discretization, then feeds the resulting DAE system to a DAE solver (a modified

version of DASSL) to integrate in time. Both the spatial and temporal error are controlled.

4.1. **B-splines.** As `BACOL` employs B-spline collocation for its spatial discretization, it may useful at this point to present a brief introduction to B-spline theory.

A spline function is an interpolating function constructed by piecing together low order polynomials satisfying certain smoothness conditions. The degree of a spline function indicates the degree of its composing polynomials. B-splines, first introduced in 1947 by Curry and Schoenberg [3], form a basis for the set of all splines. Every spline of degree p can be formed by a linear combination of B-splines of degree p.

Consider the mesh $x_0 < x_1 < \cdots < x_N$. We would define a B-spline of degree 0 as

$$B_i^0(x) = \begin{cases} 1 & x \in [x_i, x_{i+1}) \\ 0 & \text{elsewhere} \end{cases}$$

We note the following properties of B_i^0 (which hold for the higher degrees).

(1) While a discontinuous function, we note that B_i^0 is continuous from the right at all points.

(2) The support of B_i^0 (where $B_i^0 \neq 0$) the half open interval $[x_i, x_{i+1})$.

(3) $B_i^0 \geq 0$ for all x and i

(4) $\sum_{i=-\infty}^{\infty} B_i^0 = 1$ for all x

Any spline S of degree 0 can be formed by a linear combination of the splines B_i^0,

$$S = \sum_{i=-\infty}^{\infty} b_i B_i^0,$$

and we can generate higher degrees recursively according to the formula

$$B_i^p = \frac{x - x_i}{x_{i+p} - x_i} B_i^{p-1} + \frac{x_{i+p+1} - x}{x_{i+p+1} - x_{i+1}} B_{i+1}^{p-1}.$$

We notice that each new degree is based on a linear combination of B-splines of the previous degree, and our support increases with each new degree. For each B-spline of degree p, the support is composed of $p + 1$ subintervals.

4.2. **Spatial Discretization.** The following discussion of `BACOL`'s spatial discretization approach is adapted from [7]. We consider a mesh consisting of $N + 1$ points in the domain $[0, 1]$ where

$$0 = x_0 < x_1 < x_2 < \cdots < x_N = 1.$$

We introduce no loss of generality by considering the interval $[0, 1]$ as we can map any domain in our problem class onto this. For each subinterval $[x_{i-1}, x_i]$ there is an associated polynomial of degree p. We impose continuity conditions (C^1-continuity) at each internal mesh point x_i and the dimension of the piecewise polynomial subspace, S_p, becomes

$$\dim(S_p) = NP = N(p-1) + 2.$$

Wang chooses to use a B-spline basis for the polynomial subspace due to favorable condition numbers on the resulting Jacobians for parabolic PDEs, however any polynomial basis could be chosen in theory. We denote the b-spline basis for S_p by $\{B_i(x)\}_{i-1}^{NC}$. We saw in the previous section that the support for a degree p B-spline is $p+1$ subintervals. It follows that for any x in $[x_{k-1}, x_k), 1 \leq k \leq N$, at most $p+1$ B-spline basis functions have nonzero values (specifically, $\{B_i(x)\}_{i=(k-1)(p-1)+1}^{k(p-1)+2}$).

The approximate solution to our system, $U(x,t)$, can be expressed as a linear combination of B-splines. The s-th component of the solution $u_s(x,t)$ is approximated by the piecewise polynomial

$$U_s(x,t) = \sum_{i=1}^{NC} B_i(x) y_{i,s}(t),$$

where $s = 1, \ldots, NPDE$ (number of PDEs). We then collocate at the boundary points and internal Gauss points on each subinterval. These

points over the entire domain are:

$$\begin{aligned} \xi_1 &= 0 \\ \xi_l &= x_{i-1} + h_i\rho_j, \quad l = 1 + (i-1)(p-1) + j \\ \xi_{NC} &= 1, \end{aligned}$$

where $h_i = x_i - x_{i-1}$, the ρ_i are the internal Gauss points, and $i = 1, \ldots, N$,$j = 1, \ldots, p-1$. The approximation U_s is required to satisfy the PDEs at the Gauss points, resulting in $N(p-1)$ collocation equations for each PDE component:

$$\frac{d}{dt}U_s(\xi_l, t) = f(t, \xi_l, U(\xi_l, t), U_x(\xi_l, t), U_{xx}(\xi_l, t)). \tag{4.1}$$

We recall that for each Gauss point, there are at most $p+1$ nonzero B-spline basis functions, and so the linear combination is

$$U_s(\xi_l, t) = \sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} B_m(\xi_l) y_{m,s}(t).$$

We substitute this into (4.1) and get our collocation system

$$\sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} B_m(\xi_l) y'_{m,s}(t) = f_s(t, \xi_l, U(\xi_l, t), U_x(\xi_l, t), U_{xx}(\xi_l, t)),$$

where $l = 1, \ldots, NC$ and $s = 1, \ldots, NPDE$.

We are now left to consider how to treat our boundary conditions. One option would be to differentiate the BCs with respect to time and add these resulting ODEs to our system and solve with any of

the available ODE solvers. The other option, chosen in BACOL, is to leave the boundary conditions as they are, treating them as algebraic constraints. The boundary conditions, coupled with our collocated system form a differential algebraic equation (DAE) system. This DAE system is now integrated in time with a modified version of DASSL, a public domain DAE solver based on backward differentiation formulas (BDF).

4.3. **Spatial Adaptivity.** In BACOL, the spatial mesh is adapted according to error tolerance violations. With no analytic solution available for every system in our problem class, error estimates are calculated by solving the system with B-splines at degree p and also at degree $p+1$. The two systems are

$$\begin{aligned}
b_L(t, U(0,t), U_x(0,t)) &= 0, \\
\sum_{m=(i-1)(p-1)+1}^{i(p-1)+2} B_m(\xi_l) y'_{m,s}(t) &= f_s(t, \xi_l, U(\xi_l,t), U_x(\xi_l,t), U_{xx}(\xi_l,t)), \\
b_R(t, U(1,t), U_x(1,t)) &= 0, \\
\\
b_L(t, \overline{U}(0,t), \overline{U}_x(0,t)) &= 0, \\
\sum_{m=(i-1)(p-1)+1}^{ip+2} \overline{B}_m(\overline{\xi}_l) \overline{y}'_{m,s}(t) &= f_s(t, \overline{\xi}_l, \overline{U}(\overline{\xi}_l,t), \overline{U}_x(\overline{\xi}_l,t), \overline{U}_{xx}(\overline{\xi}_l,t)), \\
b_R(t, \overline{U}(1,t), \overline{U}_x(1,t)) &= 0,
\end{aligned}$$

where $l = 1, \ldots, N$ and $s = 1, \ldots, NPDE$. Rather than solve these these independently, they are given to DASSL as a single system. The error estimate is then calculated *a posteriori* with the solutions to the p and $p+1$ degree systems. the normalized error for the s-th component of the PDE system (over the whole domain) is given by

$$||E_s|| = \sqrt{\int_0^1 \left(\frac{U_s - \overline{U}_s}{ATOL_s + RTOL_s|U_s|} \right)^2 dx},$$

where $ATOL_s$ and $RTOL_s$ denote the absolute and relative tolerances for the s-th component of the system. The error in each subinterval is also calculated:

$$||\hat{E}_s|| = \sqrt{\sum_{s=1}^{NPDE} \int_0^1 \left(\frac{U_s - \overline{U}_s}{ATOL_s + RTOL_s|U_s|} \right)^2 dx},$$

where $i = 1, \ldots, N$.

With the errors in hand, the following parameters are calculated

$$\begin{aligned} r_1 &= \max_{1 \le i \le N} ||\hat{E}_i||^{\frac{1}{(p+1)}}, \\ r_2 &= \frac{\sum_{i=1}^{N} ||\hat{E}_i||^{\frac{1}{(p+1)}}}{N}. \end{aligned}$$

The maximum subinterval error is given by r_1 while the average error over the subintervals is given by r_2. If $r_1 >> r_2$ then the mesh is determined to be poorly distributed. `BACOL` checks if

$$\frac{r_1}{r_2} > 2$$

or if

$$||E|| = \max_{0 \le s \le NPDE} ||E_s|| \ge 1.$$

Should either of these conditions hold, the mesh is refined according to a global equidistribution strategy (see [7] for details).

5. Conclusions and Future Work

5.1. **Results.** We solve the PDE system in `BACOL` with the constants set to anode values as prescribed by Stockie in [5] and the error tolerance (both absolute and relative for all PDE components) set to 10^{-8}. These are the same values used in the Matlab implementation. The driving programs and user supplied subroutines can be found in the Appendix. We initialize `BACOL` with a spatial grid of 32 points in $[0, H]$ but `BACOL` will end up adapting this grid as solution stiffness dictates. The final temporal mesh is not available as output from `BACOL` so for our comparison in time, we read in the temporal mesh used in the Stockie's Matlab program and call `BACOL` once for each time it contains. Changing the `IDID` flag keeps `BACOL` from repeating any work done. In this way, we are able to compare results of both implementations at the same times (Figure (4)). We see that the mole fractions (R/C) and mixture concentrations, C, are similar but do have a somewhat significant difference in the sloping portion of the curve. This difference

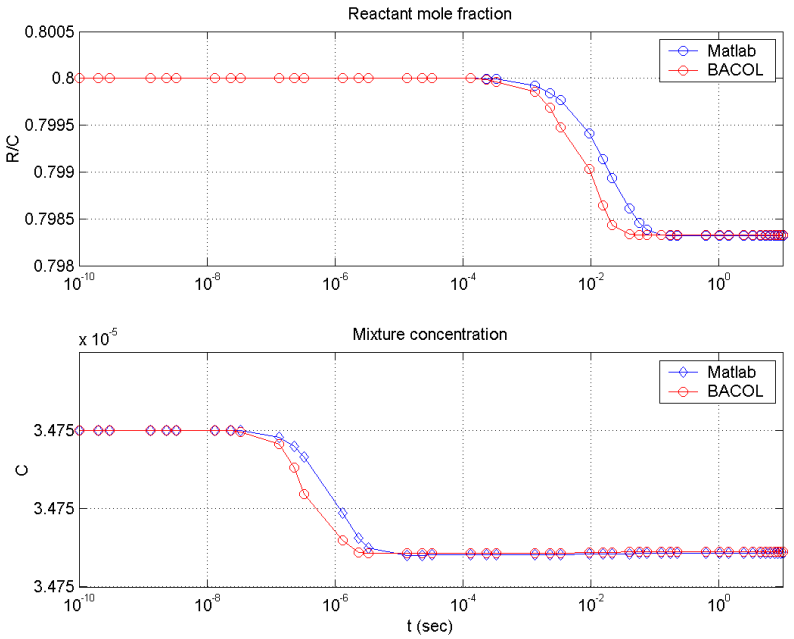

FIGURE 4. `BACOL` versus Matlab implementation

may be due to increased accuracy achieved by using an adaptive mesh code rather than a fixed mesh code but without actual fuel cell data to compare both against, it is difficult to reach any firm conclusion. Runs of the Matlab code with the spatial mesh changed from 32 points in the domain to 10000 points in the domain changed the result only mildly, and certainly not enough to account for the difference between `BACOL` and Matlab. This issue requires further study.

Both solution components in time are shown in Figure (5) and the shape of the solution in both space and time is shown in Figure (6). The spatial shape compares well with the relevant figures in [4].

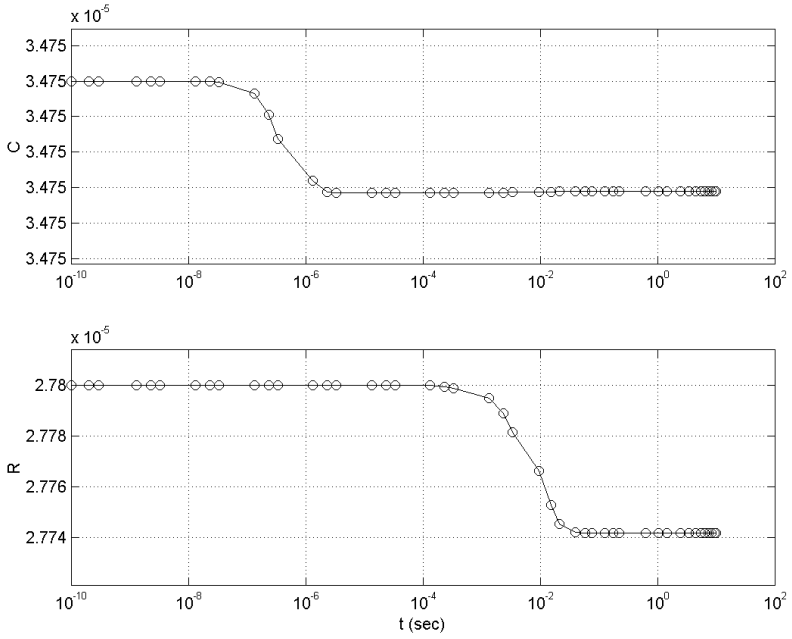

FIGURE 5. Components of solution in time (C and R)

5.2. **Convergence Analysis.** In order to test the convergence of the solution as the error tolerance is decreased, we run `BACOL` with error tolerances from 10^{-6} to 10^{-14}. These curves are plotted in Figure (7). We see some significant deviation where the steep portion of the solution flattens out (from $t = 0.01$ to $t = 0.1$) as the tolerance drops from 10^{-6} to 10^{-9} but the stricter tolerances are nearly indistinguishable. This behavior is depicted in Figures (8) and (9) which show the average (over space and time) of the absolute change in the solution as the tolerance decreases. For both solution components, we see that we

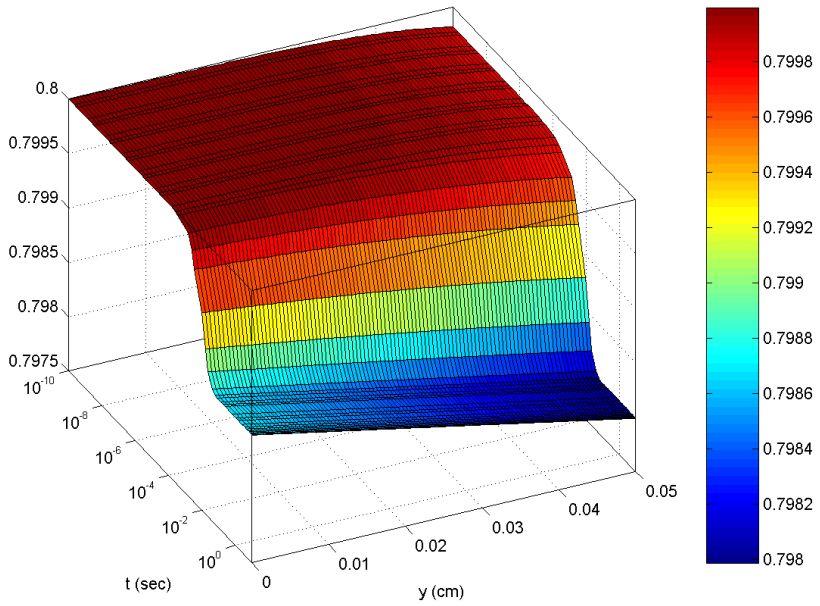

FIGURE 6. Reactant mole fraction (R/C) in space and time

converge (for higher tolerances) to a single solution as the tolerance decreases. We note that floating point exceptions (underflow) are raised for the tolerances 10^{-13} and 10^{-14}.

5.3. **Project Goals.** My initial goals for this project were to implement Stockie's discretized model in Matlab (indpendent of Stockie's own Matlab implementation), validate the results and, should time allow, add another level of complexity to the model. My initial impulse was to use Matlab's `bvp4c` however, it proved exceedingly difficult to

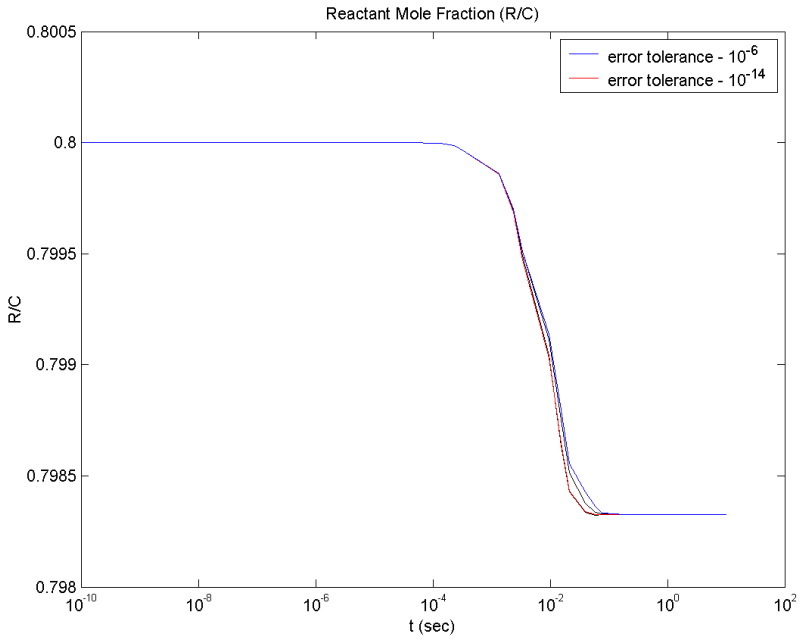

FIGURE 7. Convergence Analysis. Reactant mole fractions versus time are plotted for error tolerances of $10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-10}, 10^{-11}, 10^{-12}, 10^{-13}$, and 10^{-14}

configure the problem to fit the solver, especially since we were imposing a fixed spatial mesh by our discretization. Upon Dr. Spiteri's suggestion, I solved the PDE system outright with Rong Wang's `BACOL` software. Without actual data to test against, I validated the results by noting the relative agreement with the results from Stockie's Matlab code. Adding additional complexity to the model within the time frame of this project proved overly optimistic, however we now have

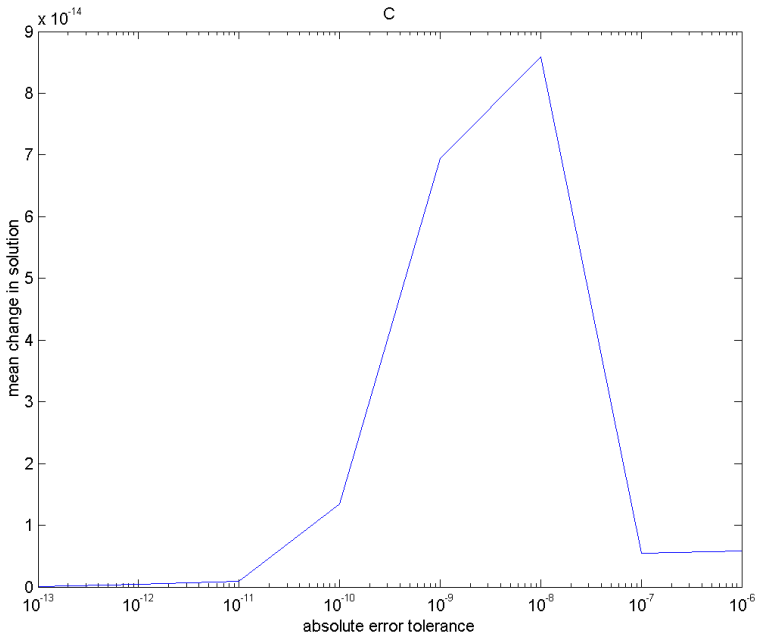

FIGURE 8. Change in C with decreasing tolerance

a solid base from which we can work towards adding a condensation component or another spatial dimension to the model.

5.4. **Future Work.** There is much potential with the 1-d fuel cell problem for future developments.

5.4.1. *Validation.* A desirable next step would be to obtain some benchmark (perhaps real data) against which we can compare both `BACOL`'s and Stockie's Matlab results. This would tell us if the `BACOL` implementation is indeed an improvement as suspected, or if there is more

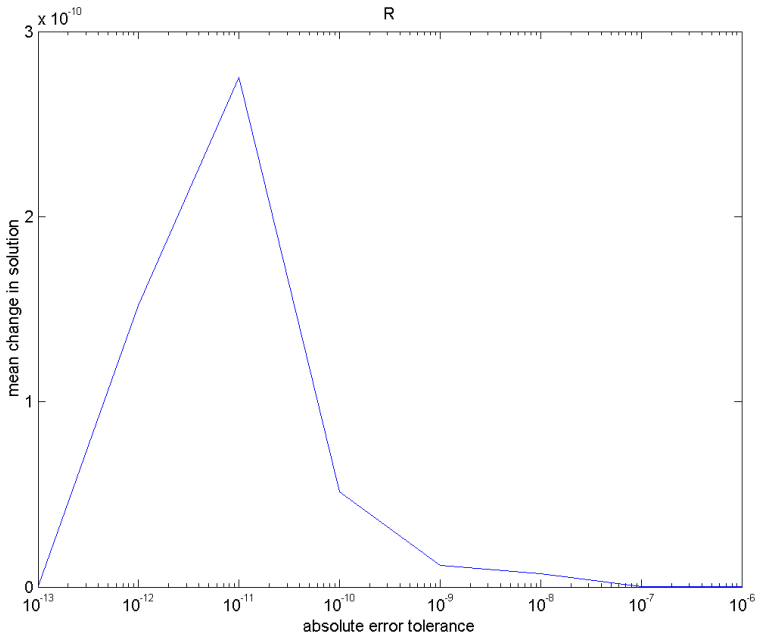

FIGURE 9. Change in R with decreasing tolerance

work to do to get improved results. So far, the most we can say about our results is that they are credible.

5.4.2. *Condensation.* Once we are confident in the `BACOL` implementation, Stockie suggests adding condensation to the 1-d model. Recall that one of the advantages of fuel cells is the ability to operate at relatively low temperatures. As the operating temperature falls, we begin to see the involved gases condensing (especially the resulting water vapour). We now have to concern ourselves not just with a multicomponent gas, but a multicomponent, multiphase mixture. Stockie

develops a two-dimensional model that includes condensation in [6]. It would be desirable to have a simple 1-d model that could more easily couple with other fuel cell components.

5.4.3. *2-Dimensional Problem.* Another direction this research could take has been suggested by Dr. Paul Muir in conversations with Dr. Spiteri. A method of lines discretization could be done by hand in the other spatial dimension with some periodic boundary conditions and each component of the resulting system solved with `BACOL`. This 2-d model could then be tested against other 2-d approaches. We would also need to determine if this approach using `BACOL` would yield any advantages over using a 2-d PDE solver, such as VLUGR2 by Blom and Verwer [2], outright.

5.4.4. *Modification of* `BACOL`. One of the main advantages of `BACOL` is that it is free to develop it's own mesh in space and time. Studying the resulting meshes can often yield useful information about the nature of the solution and exactly where the difficult areas lie. Currently, `BACOL` only returns the spatial mesh it determines when solving the problem. Introducing the option of outputting the temporal mesh would be a useful modification, especially for the fuel cell problem. This would likely involve further modification of the time integrator DASSL to

obtain the step sizes which, in private communication, Wang cites as "very hard, even for the advanced user".

6. Appendix

6.1. **BACOL Driving Program.**

```
C-----------------------------------------------------------------------
C CONSTANTS:
        INTEGER                 KCOL
C                               KCOL IS THE NUMBER OF COLLOCATION POINTS
C                               TO BE USED IN EACH SUBINTERVAL, WHICH IS
C                               EQUAL TO THE DEGREE OF THE PIECEWISE
C                               POLYNOMIALS MINUS ONE.
C                               1 < KCOL < 11.
        PARAMETER               (KCOL = 3)
        INTEGER                 NPDE
C                               NUMBER OF PDES
        PARAMETER               (NPDE = 2)
        INTEGER                 NINTMX
C                               MAXIMAL NUMBER OF INTEVALS ALLOWED
        PARAMETER               (NINTMX = 2000)
        INTEGER                 MAXVEC
C                               THE DIMENSION OF THE VECTOR OF
C                               BSPLINE COEFFICIENTS
        PARAMETER               (MAXVEC = NPDE*(NINTMX*KCOL+2))
        INTEGER                 LRP
C                               SEE THE COMMENT FOR RPAR
        PARAMETER               (LRP =134+NINTMX*(35+35*KCOL+31*NPDE
     +                               +38*NPDE*KCOL+8*KCOL*KCOL)+14*KCOL
     +                               +79*NPDE+NPDE*NPDE*(21
     +                               +4*NINTMX*KCOL*KCOL+12*NINTMX*KCOL
     +                               +6*NINTMX))
C
        INTEGER                 LIP
C                               SEE THE COMMENT FOR IPAR
        PARAMETER               (LIP = 115+NPDE*((2*KCOL+1)*NINTMX+4))
        INTEGER                 LENWRK
C                               THE DIMENSION OF ARRAY WORK WHEN WE
C                               CALL VALUES
        PARAMETER               (LENWRK =(KCOL+2)+KCOL*(NINTMX+1)+4)
        INTEGER                 NUOUT
C                               THE DIMENSION OF UOUT
        PARAMETER               (NUOUT = NPDE*101)
        DOUBLE PRECISION        XA
C                               THE LEFT BOUNDARY POINT
        PARAMETER               (XA = 0.0D0)
        DOUBLE PRECISION        XB
C                               THE RIGHT BOUNDARY POINT
        PARAMETER               (XB = 0.05D0)
C-----------------------------------------------------------------------

        DOUBLE PRECISION        T0, TVEC(45)
C                               T0 < TOUT IS THE INITIAL TIME.
C
        DOUBLE PRECISION        TOUT
C                               TOUT IS THE DESIRED FINAL OUTPUT TIME.
C
        DOUBLE PRECISION        ATOL(NPDE)
C                               ATOL IS THE ABSOLUTE ERROR TOLERANCE
```

```
C                               REQUEST AND IS A SCALAR QUANTITY IF
C                               MFLAG(2) = 0.
C
      DOUBLE PRECISION          RTOL(NPDE)
C                               RTOL IS THE RELATIVE ERROR TOLERANCE
C                               REQUEST AND IS A SCALAR QUANTITY IF
C                               MFLAG(2) = 0.
C
      INTEGER                   NINT
C                               NINT IS THE NUMBER OF SUBINTERVALS
C                               DEFINED BY THE SPATIAL MESH X.
C
      DOUBLE PRECISION          X(NINTMX+1)
C                               X IS THE SPATIAL MESH WHICH DIVIDES THE
C                               INTERVAL [X_A,X_B] AS: X_A = X(1) <
C                               X(2) < X(3) < ... < X(NINT+1) = X_B.
C
      INTEGER                   MFLAG(5)
C                               THIS VECTOR OF USER INPUT DETERMINES
C                               THE INTERACTION OF BACOL WITH DASSL.
C
C     WORK STORAGE:
      DOUBLE PRECISION          RPAR(LRP)
C                               RPAR IS A FLOATING POINT WORK ARRAY
C                               OF SIZE LRP.
C
      INTEGER                   IPAR(LIP)
C                               IPAR IS AN INTEGER WORK ARRAY
C                               OF SIZE LIP.
C
C-----------------------------------------------------------------------
      DOUBLE PRECISION          Y(MAXVEC)
C                               ON SUCCESSFUL RETURN FROM BACOL, Y IS
C                               THE VECTOR OF BSPLINE
C                               COEFFICIENTS AT THE CURRENT TIME TO.
C
      INTEGER                   IDID
C                               IDID IS THE BACOL EXIT STATUS FLAG
C                               WHICH IS BASED ON THE EXIT STATUS FROM
C                               DASSL ON ERROR CHECKING PERFORMED BY
C                               BACOL ON INITIALIZATION.
C-----------------------------------------------------------------------
      DOUBLE PRECISION          UOUT(NUOUT)
C                               THE APPROXIMATION SOLUTIONS AT A SET
C                               OF POINTS
      DOUBLE PRECISION          VALWRK(LENWRK)
C                               VALWRK IS A WORK ARRAY IN VALUES
      DOUBLE PRECISION          XOUT(101)
C                               XOUT IS A SET OF SPATIAL POINTS FOR
C                               OUTPUT
      DOUBLE PRECISION          GAMMA, D, NU, RO, RH, CBAR, RBAR
C                               MODEL PARAMETERS
      COMMON /FUEL/             GAMMA, D, NU, RO, RH, CBAR, RBAR
      INTEGER                   I, J
C-----------------------------------------------------------------------
C SUBROUTINES CALLED:
C                               BACOL
C                               VALUES
```

```
C-----------------------------------------------------------------------

C     SET THE REMAINING INPUT PARAMETERS.
      T0 = 0.0D0

      open(83,file='times.dat')
      do 9 I=1,45
     READ(83,*) TVEC(I)
    9 continue
      close(83)

      print *, TVEC

      TOUT = 5.0D-4
      ATOL(1) = 1.D-8
      RTOL(1) = ATOL(1)
      NINT = 32
C
C     SET THE MODEL PARAMETERS (as suggested by Stockie)

      GAMMA = 3.24d7            ! CONVECTION
      D     = 0.29d0            ! DIFFUSIVITY
      NU    = 1.00d0                    ! RETURN COEFFICIENT
      R0    = 3.0                       ! BOTTOM TRANSFER RATE
      RH    = 0.005                     ! TOP TRANSFER RATE
      CBAR  = 3.475d-5                  ! MIXTURE CONCENTRATION
      RBAR  = 2.780d-5                  ! REACTANT CONCENTRATION


C     DEFINE THE MESH BASED ON A UNIFORM STEP SIZE.
      X(1) = XA
      DO 10 I = 2, NINT
         X(I) = XA + ((I-1) * (XB - XA)) / NINT
   10 CONTINUE
      X(NINT+1) = XB

C     INITIALIZE THE MFLAG VECTOR.
      DO 20 I = 1, 5
         MFLAG(I) = 0
   20 CONTINUE

      WRITE(6,'(/A, I3, A, I4, 2(A, E8.2))') 'KCOL =', KCOL, ' NINT =',
     &  NINT, ' ATOL(1) =', ATOL(1), ' RTOL(1) =', RTOL(1)
      WRITE(6, '(/A, E8.2)') 'GAMMA =', GAMMA

      XOUT(1) = XA
      DO 30 I = 2, 100
         XOUT(I) = XA + DBLE(I - 1) * (XB - XA)/100.D0
   30 CONTINUE
      XOUT(101) = XB

      open(85,file='fuel.txt')

      do 999 j = 1,45
      CALL BACOL(T0, TOUT, ATOL, RTOL, NPDE, KCOL, NINTMX, NINT, X,
     &           MFLAG, RPAR, LRP, IPAR, LIP, Y, IDID)

C     CHECK FOR AN ERROR FROM BACOL.
```

```
C     WRITE(6,'(A, I5)') 'IDID =', IDID
      IF (IDID .LT. 2) GOTO 100

      CALL VALUES(KCOL,XOUT,NINT,X,NPDE,101,0,1,UOUT,Y,VALWRK)

      do 990 i=1,101
      WRITE(85,*) i, UOUT(1+2*(i-1)), UOUT(i*2)
  990 continue
     if (j.lt.45) then
           tout = tout + TVEC(j+1)
           idid = 1
     endif
  999 continue
      close(85)
      GOTO 9999

  100 CONTINUE
      WRITE(6,'(A)') 'CANNOT PROCEED DUE TO ERROR FROM BACOL.'

 9999 STOP
      END
```

6.2. **Fuel Cell Problem Subroutines.**

```
C-----------------------------------------------------------------------
      SUBROUTINE F(T, X, U, UX, UXX, FVAL, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THIS SUBROUTINE DEFINES THE RIGHT HAND SIDE VECTOR OF THE
C       NPDE DIMENSIONAL PARABOLIC PARTIAL DIFFERENTIAL EQUATION
C                         UT = F(T, X, U, UX, UXX).
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
        INTEGER                 NPDE
C                               THE NUMBER OF PDES IN THE SYSTEM.
C
        DOUBLE PRECISION        T
C                               THE CURRENT TIME COORDINATE.
C
        DOUBLE PRECISION        X
C                               THE CURRENT SPATIAL COORDINATE.
C
        DOUBLE PRECISION        U(NPDE)
C                               U(1:NPDE) IS THE APPROXIMATION OF THE
C                               SOLUTION AT THE POINT (T,X).
C
        DOUBLE PRECISION        UX(NPDE)
C                               UX(1:NPDE) IS THE APPROXIMATION OF THE
C                               SPATIAL DERIVATIVE OF THE SOLUTION AT
C                               THE POINT (T,X).
C
        DOUBLE PRECISION        UXX(NPDE)
C                               UXX(1:NPDE) IS THE APPROXIMATION OF THE
C                               SECOND SPATIAL DERIVATIVE OF THE
C                               SOLUTION AT THE POINT (T,X).
```

```
C
C OUTPUT:
        DOUBLE PRECISION        FVAL(NPDE)
C                               FVAL(1:NPDE) IS THE RIGHT HAND SIDE
C                               VECTOR F(T, X, U, UX, UXX) OF THE PDE.
C-----------------------------------------------------------------------
      DOUBLE PRECISION GAMMA, D, NU, R0, RH, CBAR, RBAR
      COMMON /FUEL/ GAMMA, D, NU, R0, RH, CBAR, RBAR
C-----------------------------------------------------------------------
C
C     ASSIGN FVAL(1:NPDE) ACCORDING TO THE RIGHT HAND SIDE OF THE PDE
C     IN TERMS OF U(1:NPDE), UX(1:NPDE), UXX(1:NPDE).
C


      FVAL(1) = GAMMA*UX(1)**2 + GAMMA*U(1)*UXX(1)

      FVAL(2) = GAMMA*UX(2)*UX(1) + GAMMA*U(2)*UXX(1) +
     &          D*UX(1)*( UX(2)/U(1) - U(2)*UX(1)/(U(1)**2) ) +
     &          D*U(1)*( UXX(2)/U(1) - 2.d0*UX(2)*UX(1)/(U(1)**2) +
     &          2.d0*U(2)*UX(1)**2/(U(1)**3) - U(2)*UXX(1)/(U(1)**2) )
C
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE DERIVF(T, X, U, UX, UXX, DFDU, DFDUX, DFDUXX, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THIS SUBROUTINE IS USED TO DEFINE THE INFORMATION ABOUT THE
C       PDE REQUIRED TO FORM THE ANALYTIC JACOBIAN MATRIX FOR THE DAE
C       OR ODE SYSTEM. ASSUMING THE PDE IS OF THE FORM
C                        UT = F(T, X, U, UX, UXX)
C       THIS ROUTINE RETURNS THE JACOBIANS D(F)/D(U), D(F)/D(UX), AND
C       D(F)/D(UXX).
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
        INTEGER                 NPDE
C                               THE NUMBER OF PDES IN THE SYSTEM.
C
        DOUBLE PRECISION        T
C                               THE CURRENT TIME COORDINATE.
C
        DOUBLE PRECISION        X
C                               THE CURRENT SPATIAL COORDINATE.
C
        DOUBLE PRECISION        U(NPDE)
C                               U(1:NPDE) IS THE APPROXIMATION OF THE
C                               SOLUTION AT THE POINT (T,X).
C
        DOUBLE PRECISION        UX(NPDE)
C                               UX(1:NPDE) IS THE APPROXIMATION OF THE
C                               SPATIAL DERIVATIVE OF THE SOLUTION AT
C                               THE POINT (T,X).
C
        DOUBLE PRECISION        UXX(NPDE)
C                               UXX(1:NPDE) IS THE APPROXIMATION OF THE
```

```
C                               SECOND SPATIAL DERIVATIVE OF THE
C                               SOLUTION AT THE POINT (T,X).
C
C OUTPUT:
        DOUBLE PRECISION        DFDU(NPDE,NPDE)
C                               DFDU(I,J) IS THE PARTIAL DERIVATIVE
C                               OF THE I-TH COMPONENT OF THE VECTOR F
C                               WITH RESPECT TO THE J-TH COMPONENT
C                               OF THE UNKNOWN FUNCTION U.
C
        DOUBLE PRECISION        DFDUX(NPDE,NPDE)
C                               DFDUX(I,J) IS THE PARTIAL DERIVATIVE
C                               OF THE I-TH COMPONENT OF THE VECTOR F
C                               WITH RESPECT TO THE J-TH COMPONENT
C                               OF THE SPATIAL DERIVATIVE OF THE
C                               UNKNOWN FUNCTION U.
C
        DOUBLE PRECISION        DFDUXX(NPDE,NPDE)
C                               DFDUXX(I,J) IS THE PARTIAL DERIVATIVE
C                               OF THE I-TH COMPONENT OF THE VECTOR F
C                               WITH RESPECT TO THE J-TH COMPONENT
C                               OF THE SECOND SPATIAL DERIVATIVE OF THE
C                               UNKNOWN FUNCTION U.
      DOUBLE PRECISION GAMMA, D, NU, RO, RH, CBAR, RBAR
      COMMON /FUEL/ GAMMA, D, NU, RO, RH, CBAR, RBAR
C-----------------------------------------------------------------------
C
C     ASSIGN DFDU(1:NPDE,1:NPDE), DFDUX(1:NPDE,1:NPDE), AND
C     DFDUXX(1:NPDE,1:NPDE) ACCORDING TO THE RIGHT HAND SIDE OF THE PDE
C     IN TERMS OF U(1:NPDE), UX(1:NPDE), UXX(1:NPDE).
C
      DFDU(1,1)   = GAMMA*UXX(1)
      DFDUX(1,1)  = 2.d0*GAMMA*UX(1)
      DFDUXX(1,1) = GAMMA*U(1)

      DFDU(1,2)   = 0.d0
      DFDUX(1,2)  = 0.d0
      DFDUXX(1,2) = 0.d0

      DFDU(2,1) = D*UX(1)*(-UX(2)/(U(1)**2)+2.d0*U(2)*UX(1)/(U(1)**3))
     &          + D*(UXX(2)/U(1)-2.d0*UX(2)*UX(1)/(U(1)**2)
     &          + 2.d0*U(2)*UX(1)**2/(U(1)**3)-U(2)*UXX(1)/(U(1)**2))
     &          + D*U(1)*(-UXX(2)/(U(1)**2)+4.d0*UX(2)*UX(1)/(U(1)**3)
     &          - 6.d0*U(2)*UX(1)**2/(U(1)**4)
     &          + 2.d0*U(2)*UXX(1)/(U(1)**3) )

      DFDUX(2,1) = GAMMA*UX(2)+D*(UX(2)/U(1)-U(2)*UX(1)/(U(1)**2))
     &           - D*UX(1)*U(2)/(U(1)**2)+D*U(1)*(-2.d0*UX(2)/(U(1)**2)
     &           + 4.d0*U(2)*UX(1)/(U(1)**3) )

      DFDUXX(2,1) = GAMMA*U(2) - D*U(2)/U(1)

      DFDU(2,2)   = GAMMA*UXX(1)-D*UX(1)**2/(U(1)**2)
     &        + D*U(1)*(2.d0*UX(1)**2/(U(1)**3)-UXX(1)/(U(1)**2))
      DFDUX(2,2)  = GAMMA*UX(1)-D*UX(1)/U(1)

      DFDUXX(2,2) = D
C
```

```
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE BNDXA(T, U, UX, BVAL, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THE SUBROUTINE IS USED TO DEFINE THE BOUNDARY CONDITIONS AT THE
C       LEFT SPATIAL END POINT X = XA.
C                           B(T, U, UX) = 0
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
        INTEGER                 NPDE
C                               THE NUMBER OF PDES IN THE SYSTEM.
C
        DOUBLE PRECISION        T
C                               THE CURRENT TIME COORDINATE.
C
        DOUBLE PRECISION        U(NPDE)
C                               U(1:NPDE) IS THE APPROXIMATION OF THE
C                               SOLUTION AT THE POINT (T,XA).
C
        DOUBLE PRECISION        UX(NPDE)
C                               UX(1:NPDE) IS THE APPROXIMATION OF THE
C                               SPATIAL DERIVATIVE OF THE SOLUTION AT
C                               THE POINT (T,XA).
C
C OUTPUT:
        DOUBLE PRECISION        BVAL(NPDE)
C                               BVAL(1:NPDE) IS THE BOUNDARY CONTIDITION
C                               AT THE LEFT BOUNDARY POINT.
C-----------------------------------------------------------------------
      DOUBLE PRECISION GAMMA, D, NU, RO, RH, CBAR, RBAR
      COMMON /FUEL/ GAMMA, D, NU, RO, RH, CBAR, RBAR
C-----------------------------------------------------------------------

      BVAL(1) = U(1)-CBAR
      BVAL(2) = UX(2)/U(1) - U(2)*UX(1)/(U(1)**2)
     &        + RO*(RBAR-U(2))/(D*CBAR)
C
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE BNDXB(T, U, UX, BVAL, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THE SUBROUTINE IS USED TO DEFINE THE BOUNDARY CONDITIONS AT THE
C       RIGHT SPATIAL END POINT X = XB.
C                           B(T, U, UX) = 0
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
        INTEGER                 NPDE
C                               THE NUMBER OF PDES IN THE SYSTEM.
C
        DOUBLE PRECISION        T
```

```
C                               THE CURRENT TIME COORDINATE.
C
        DOUBLE PRECISION        U(NPDE)
C                               U(1:NPDE) IS THE APPROXIMATION OF THE
C                               SOLUTION AT THE POINT (T,XB).
C
        DOUBLE PRECISION        UX(NPDE)
C                               UX(1:NPDE) IS THE APPROXIMATION OF THE
C                               SPATIAL DERIVATIVE OF THE SOLUTION AT
C                               THE POINT (T,XB).
C
C OUTPUT:
        DOUBLE PRECISION        BVAL(NPDE)
C                               BVAL(1:NPDE) IS THE BOUNDARY CONTIDITION
C                               AT THE RIGHT BOUNDARY POINT.
C-----------------------------------------------------------------------
      DOUBLE PRECISION GAMMA, D, NU, RO, RH, CBAR, RBAR
      COMMON /FUEL/ GAMMA, D, NU, RO, RH, CBAR, RBAR
C-----------------------------------------------------------------------
      BVAL(1) = UX(2)/U(1) - U(2)*UX(1)/(U(1)**2)+ RH*U(2)/(D*U(1))
      BVAL(2) = UX(1) + NU*RH*U(2)/( GAMMA*( U(1)-NU*U(2) ) )
C
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE DIFBXA(T, U, UX, DBDU, DBDUX, DBDT, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THE SUBROUTINE IS USED TO DEFINE THE DIFFERENTIATED BOUNDARY
C       CONDITIONS AT THE LEFT SPATIAL END POINT X = XA. FOR THE
C       BOUNDARY CONDITION EQUATION
C                               B(T, U, UX) = 0
C       THE PARTIAL DERIVATIVES DB/DU, DB/DUX, AND DB/DT ARE SUPPLIED
C       BY THIS ROUTINE.
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
        INTEGER                 NPDE
C                               THE NUMBER OF PDES IN THE SYSTEM.
C
        DOUBLE PRECISION        T
C                               THE CURRENT TIME COORDINATE.
C
        DOUBLE PRECISION        U(NPDE)
C                               U(1:NPDE) IS THE APPROXIMATION OF THE
C                               SOLUTION AT THE POINT (T,X).
C
        DOUBLE PRECISION        UX(NPDE)
C                               UX(1:NPDE) IS THE APPROXIMATION OF THE
C                               SPATIAL DERIVATIVE OF THE SOLUTION AT
C                               THE POINT (T,X).
C
C OUTPUT:
        DOUBLE PRECISION        DBDU(NPDE,NPDE)
C                               DBDU(I,J) IS THE PARTIAL DERIVATIVE
C                               OF THE I-TH COMPONENT OF THE VECTOR B
C                               WITH RESPECT TO THE J-TH COMPONENT
```

```
C                               OF THE UNKNOWN FUNCTION U.
C
        DOUBLE PRECISION        DBDUX(NPDE,NPDE)
C                               DBDUX(I,J) IS THE PARTIAL DERIVATIVE
C                               OF THE I-TH COMPONENT OF THE VECTOR B
C                               WITH RESPECT TO THE J-TH COMPONENT
C                               OF THE SPATIAL DERIVATIVE OF THE
C                               UNKNOWN FUNCTION U.
C
        DOUBLE PRECISION        DBDT(NPDE)
C                               DBDT(I) IS THE PARTIAL DERIVATIVE
C                               OF THE I-TH COMPONENT OF THE VECTOR B
C                               WITH RESPECT TO TIME T.
      DOUBLE PRECISION GAMMA, D, NU, RO, RH, CBAR, RBAR
      COMMON /FUEL/ GAMMA, D, NU, RO, RH, CBAR, RBAR
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C
C     ASSIGN DBDU(1:NPDE,1:NPDE), DBDU(1:NPDE,1:NPDE), AND DBDT(1:NPDE)
C     ACCORDING TO THE RIGHT BOUNDARY CONDITION EQUATION IN TERMS OF
C     U(1:NPDE), UX(1:NPDE), UXX(1:NPDE).
C
      DBDU(1,1)  = 1
      DBDUX(1,1) = 0

      DBDU(1,2)  = 0
      DBDUX(1,2) = 0

      DBDU(2,1)  = -UX(2)/(U(1)**2)+2.d0*U(2)*UX(1)/(U(1)**3)
      DBDUX(2,1) = -U(2)/(U(1)**2)

      DBDU(2,2)  = -UX(1)/U(1)**2 - RO/( D*CBAR )
      DBDUX(2,2) = 1/U(1)

      DBDT(1)    = 0
      DBDT(2)    = 0
C
      RETURN
      END
C-----------------------------------------------------------------------
      SUBROUTINE DIFBXB(T, U, UX, DBDU, DBDUX, DBDT, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THE SUBROUTINE IS USED TO DEFINE THE DIFFERENTIATED BOUNDARY
C       CONDITIONS AT THE RIGHT SPATIAL END POINT 1 = XB. FOR THE
C       BOUNDARY CONDITION EQUATION
C                              B(T, U, UX) = 0
C       THE PARTIAL DERIVATIVES DB/DU, DB/DUX, AND DB/DT ARE SUPPLIED
C       BY THIS ROUTINE.
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
        INTEGER                 NPDE
C                               THE NUMBER OF PDES IN THE SYSTEM.
C
        DOUBLE PRECISION        T
```

```
C                               THE CURRENT TIME COORDINATE.
C
        DOUBLE PRECISION        U(NPDE)
C                               U(1:NPDE) IS THE APPROXIMATION OF THE
C                               SOLUTION AT THE POINT (T,X).
C
        DOUBLE PRECISION        UX(NPDE)
C                               UX(1:NPDE) IS THE APPROXIMATION OF THE
C                               SPATIAL DERIVATIVE OF THE SOLUTION AT
C                               THE POINT (T,X).
C
C OUTPUT:
        DOUBLE PRECISION        DBDU(NPDE,NPDE)
C                               DBDU(I,J) IS THE PARTIAL DERIVATIVE
C                               OF THE I-TH COMPONENT OF THE VECTOR B
C                               WITH RESPECT TO THE J-TH COMPONENT
C                               OF THE UNKNOWN FUNCTION U.
C
        DOUBLE PRECISION        DBDUX(NPDE,NPDE)
C                               DBDUX(I,J) IS THE PARTIAL DERIVATIVE
C                               OF THE I-TH COMPONENT OF THE VECTOR B
C                               WITH RESPECT TO THE J-TH COMPONENT
C                               OF THE SPATIAL DERIVATIVE OF THE
C                               UNKNOWN FUNCTION U.
C
        DOUBLE PRECISION        DBDT(NPDE)
C                               DBDT(I) IS THE PARTIAL DERIVATIVE
C                               OF THE I-TH COMPONENT OF THE VECTOR B
C                               WITH RESPECT TO TIME T.
      DOUBLE PRECISION GAMMA, D, NU, RO, RH, CBAR, RBAR
      COMMON /FUEL/ GAMMA, D, NU, RO, RH, CBAR, RBAR
C-----------------------------------------------------------------------

C-----------------------------------------------------------------------
C
C     ASSIGN DBDU(1:NPDE,1:NPDE), DBDU(1:NPDE,1:NPDE), AND DBDT(1:NPDE)
C     ACCORDING TO THE RIGHT BOUNDARY CONDITION EQUATION IN TERMS OF
C     U(1:NPDE), UX(1:NPDE), UXX(1:NPDE).
C
      DBDU(1,1)  = -UX(2)/(U(1)**2) + 2.d0*U(2)*UX(1)/(U(1)**3)
     &           - RH*U(2)/( D*U(1)**2 )
      DBDUX(1,1) = -U(2)/(U(1)**2)

      DBDU(1,2)  = -UX(1)/(U(1)**2) + RH/( D*U(1) )
      DBDUX(1,2) = 1/U(1)

      DBDU(2,1)  = -NU*RH*U(2)/( GAMMA*( U(1)-NU*U(2) )**2 )
      DBDUX(2,1) = 1

      DBDU(2,2)  = NU*RH/( GAMMA*( U(1)-NU*U(2) ) )
     &           + NU**2*RH*U(2)/( GAMMA*( U(1)-NU*U(2) )**2 )
      DBDUX(2,2) = 0

      DBDT(1)    = 0
      DBDT(2)    = 0
C
      RETURN
      END
```

```
C-----------------------------------------------------------------------
      SUBROUTINE UINIT(X, U, NPDE)
C-----------------------------------------------------------------------
C PURPOSE:
C       THIS SUBROUTINE IS USED TO RETURN THE NPDE-VECTOR OF INITIAL
C       CONDITIONS OF THE UNKNOWN FUNCTION AT THE INITIAL TIME T = TO
C       AT THE SPATIAL COORDINATE X.
C
C-----------------------------------------------------------------------
C SUBROUTINE PARAMETERS:
C INPUT:
        DOUBLE PRECISION        X
C                               THE SPATIAL COORDINATE.
C
        INTEGER                 NPDE
C                               THE NUMBER OF PDES IN THE SYSTEM.
C
C OUTPUT:
        DOUBLE PRECISION        U(NPDE)
C                               U(1:NPDE) IS VECTOR OF INITIAL VALUES OF
C                               THE UNKNOWN FUNCTION AT T = TO AND THE
C                               GIVEN VALUE OF X.
C-----------------------------------------------------------------------
      DOUBLE PRECISION GAMMA, D, NU, RO, RH, CBAR, RBAR
      COMMON /FUEL/ GAMMA, D, NU, RO, RH, CBAR, RBAR
C-----------------------------------------------------------------------
C
C     ASSIGN U(1:NPDE) THE INITIAL VALUES OF U(TO,X).
C

      U(1) = CBAR
      U(2) = RBAR
C
      RETURN
      END
C-----------------------------------------------------------------------
```

References

1. Ryuichi Ashino, Michihiro Nagase, and Remi Vaillancourt, *Behind and beyond the matlab ode suite*, Computers and Mathematics with Applications **1** (2000), no. 40, 491–512.
2. J. G. Blom and J. G. Verwer, *Vlugr2: A vectorized local uniform grid refinement code for pdes in 2d*, Tech. Report NM-R93xx, Centrum voor Wiskunde en Informatica, Amsterdam, 1993.

3. H. Curry and I. Schoenberg, *On spline distributions and their limits: the polya distribution functions*, Bull. Amer. Math. (1947), no. 53, 1114.

4. Keith Promislow and John Stockie, *Adiabatic relaxation of convective-diffusive gas tranport in a porous fuel cell electrode*, SIAM J. Appl. Math. **62** (2001), no. 1, 180–205.

5. John Stockie, *Multicomponent gas transport model*, communication with Dr. R. Spiteri, March 2001.

6. ———, *Multi-phase flow and condensation in proton exchange membrane fuel cells*, Proceedings of IMECE'02 (New Orleans, Louisiana, USA), 2002 ASME International Mechanical Engineering Congress and Exposition, ASME, November 17-22 2002.

7. Rong Wang, *High order adaptive collocation software for 1-d parabolic pdes*, Phd thesis, Dalhousie University, Halifax, Nova Scotia, August 2002.

8. William Wangard, David S. Dandy, and Brian J. Miller, *A numerically stable method for integration of the multicomponent species diffusion equations*, Journal of Computational Physics (2001), no. 174, 460–472.

Dalhousie University, Halifax, Nova Scotia

E-mail address: lukeman@phys.ocean.dal.ca