Modeling Local and Advective Diffusion of Fuel Vapors to Understand Aqueous Foams in Fighting Fires

December 2011 Progress Report

Author:
Andrew Brandon
asbrando@math.umd.edu

Advisor:
Dr. Ramagopal Ananth
Naval Research Laboratory, Washington, D.C.
ramagopal.ananth@nrl.mil

Abstract

The purpose of this project is to model the diffusion of fuel vapors through both aqueous film surfaces and aqueous foam surfaces. Aqueous foams are currently being employed to combat fuel pool fires. Once an aqueous foam is applied to a fuel pool fire, an aqueous film forms on the surface of the liquid hydrocarbon pool due to the liquid drainage of the foam. It is the film that is responsible for the suppression of the fuel vapors. Experiments by Leonard [1] and Williams [2] have shown that the film's suppression of fuel vapors is not constant over time. It has become clear that by some process, fuel vapors are able to diffuse through the film and foam layers. This presents a hazard because the fuel vapors above the foam layer can re-ignite the fuel pool fire. A model is being created that simulates local and advective diffusion of fuel vapors over time until a steady state is reached. The diffusion coefficient's space will be explored in an attempt to match the model's concentration to the observed steady state concentrations in [1] and [2]. This will allow us to calculate the diffusion coefficient of fuel vapors in both film and foam layers. At this point in the project, the solvers have been programmed. Several test cases are being explored using a simple axisymmetric cylindrical container in an effort to debug the solvers. Once debugged, the results will be compared to the analytical solutions for specialized cases. Then the remaining features of the experimental domain from [1] and [2] will be added to the model domain.

1 Background

To combat fuel pool fires, the United States Navy employs several types of fluorinated fire fighting foams. Once applied to a pool fire, a portion of the liquid in the foam drains over a short period of time, depositing a layer of film on top of the fuel pool. The fluorine surfactant in the foam is responsible for decreasing the surface tension of the foam solution, allowing the aqueous layer to float on the surface of the fuel pool. This layer of film on the surface of the fuel pool suppresses further evaporation of the fuel. However, several experiments have shown that this suppression of fuel vapors is not constant over time.

In the 1970s, Dr. Joseph Leonard et al at the Naval Research Laboratory (NRL) in Washington, D.C. began testing the suppression ability of the then new aqueous film forming foam (AFFF) [1]. To test AFFF, they took the film that the foam creates and placed a specified amount on a fuel pool. Then they measured the concentration of fuel vapors over a specified amount of time. Their results confirmed that the film created by AFFF was capable of initially suppressing the vapors of several fuel types [1]. What they did not expect to find was that after the initial suppression, the concentration of fuel vapors increased with time. In some cases, fuel vapor concentrations reached levels characteristic of when foam is absent. Recently, this phenomenon has gained interest.

Dr. Bradley Williams, also of NRL, is currently working on an experiment similar to Dr. Leonard's. His experiment involves placing an aqueous foam layer on the surface of a fuel pool rather than the film layer only. When measuring fuel vapor concentration levels over a period of time, Dr. Williams has also found that the concentration of fuel vapor increases with time after the initial suppression by the foam layer.

To motivate my project, I propose an all too realistic situation. In the case of an on-board fire involving fuel spills, an aqueous foam such as AFFF will be applied to the surface. All visually apparent flames will be put out by applying AFFF, but there may exist an ember or an unseen open flame in the vicinity of the foam layer. Suppose that in some area, away from the open flame, the foam surface is compromised. If the fuel vapor concentration above the still intact foam surface reaches a certain level, the unseen open flame may ignite a fire above the foam surface [3], which is known as "ghosting." The "ghost" flames could then travel towards the open fuel pool surface resulting in re-ignition.

Currently, the Navy is involved in "burnback" experiments which test the length of time it takes for an open flame in the vicinity of a foam covered fuel pool to induce ghosting and re-ignition [3]. The aqueous foams that the Navy employs have a short filming time as well as a short "burnback" time. This short "burnback" time is directly related to the rate at which fuel vapors are diffusing through the film and foam layers [3].

In addition to not having constant vapor suppression, these fluorinated film forming foams are environmentally unfriendly and carcinogenic. Research is currently being done in order to find a replacement, however a satisfactory one has not yet been found. Comparison tests are necessary between the current product and any possible replacements in order to

insure safety and effectiveness. In order to properly compare, the processes taking place in the current product must be understood. Thus, it is very important to design a model that can calculate the rate at which fuel vapors diffuse through the foam and film layers. This will help us to understand processes such as ghosting, but also to find replacement products.

2 The Model Equations and Algorithms

The model domain for this project will be the experimental domain used in [1] and [2]. All calculations will be done in cylindrical coordinates. It will be assumed that the domain is axisymmetric, which will reduce the problem to a two dimensional case. The problem will be divided into two domains of the aqueous film or aqueous foam layer and the remainder of the container. From this point on, the film or foam domain will be denoted domain 1 and the remainder of the container will be domain 2. (See Figure 1.) By splitting the model domain into two, we will be able to pass the top boundary values of domain 1 as a lower boundary condition for domain 2. The film or foam in domain 1 will be assumed to be a continuum.

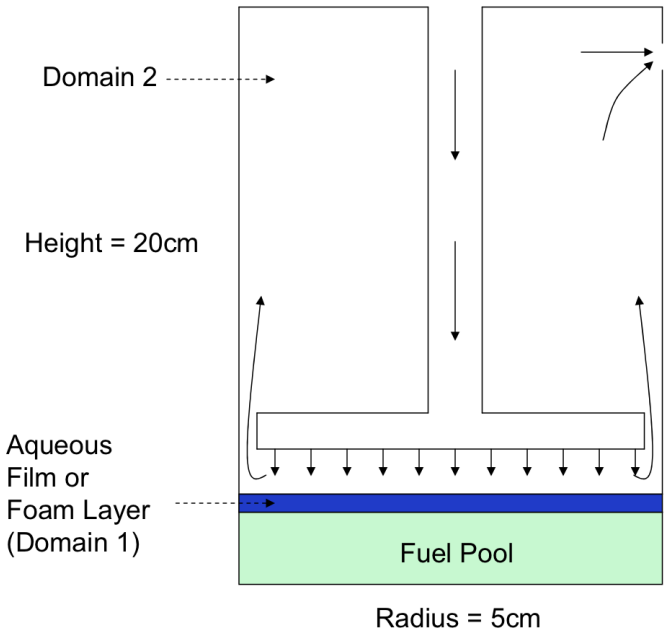

Figure 1: Experimental domain of [1] and [2] with domain 1 and domain 2.

In both Leonard's and Williams's experiments, air is pumped through a fritted glass disk, positioned two centimeters above the film or foam surface. Air then flows out of the top of the container and the fuel vapor content of that flow is analyzed. Because the air flow in [1] and [2] is slow, the foam layer will be assumed stationary. This leaves us with solving the species fraction equation (1) for both domains and the momentum equations (2) and (3) for domain 2.

$$\frac{\partial Y}{\partial t} + \nabla \cdot (\mathbf{v}Y) = D\nabla \cdot (\nabla(Y)) \quad \mathbf{v} = (u, w) \tag{1}$$

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial r} + w\frac{\partial u}{\partial z} = -\frac{1}{\rho}\frac{\partial P}{\partial r} + \frac{\mu}{\rho}\left(\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r} + \frac{\partial^2 u}{\partial z^2}\right) \tag{2}$$

$$\frac{\partial w}{\partial t} + u\frac{\partial w}{\partial r} + w\frac{\partial w}{\partial z} = -\frac{1}{\rho}\frac{\partial P}{\partial z} + \frac{\mu}{\rho}\left(\frac{\partial^2 w}{\partial r^2} + \frac{1}{r}\frac{\partial w}{\partial r} + \frac{\partial^2 w}{\partial z^2}\right) \tag{3}$$

Y denotes the species fraction of fuel vapors [4]. The variable u represents the radial velocity and the variable w represents the axial velocity [5]. The diffusion coefficient is represented by D. In domain 2, the diffusion coefficient (D_2) of fuel vapors in air is known. In domain 1, the diffusion coefficient (D_1) is not known. We will be optimizing over D_1 to find a diffusion coefficient corresponding to the steady state data from [1] and [2]. Density will be assumed constant in both domains because the mixing of fuel vapors with air, foam, and film results in a very small change in density. This means that $\nabla \cdot (\mathbf{v}) = 0$ from the continuity equation. Thus, we can use the relation $\nabla \cdot (\mathbf{v}Y) = \mathbf{v}\nabla Y + Y\nabla \cdot (\mathbf{v})$ and simplify it to $\nabla \cdot (\mathbf{v}Y) = \mathbf{v}\nabla Y$ by applying $\nabla \cdot (\mathbf{v}) = 0$. Thus, (1) can be rewritten in cylindrical coordinates as

$$\frac{\partial Y}{\partial t} + u\frac{\partial Y}{\partial r} + w\frac{\partial Y}{\partial z} = D\left(\frac{\partial^2 Y}{\partial r^2} + \frac{1}{r}\frac{\partial Y}{\partial r} + \frac{\partial^2 Y}{\partial z^2}\right) \tag{4}$$

In order to solve for u and w, the substitution $u = \frac{-1}{r}\frac{\partial \psi}{\partial z}$ and $w = \frac{1}{r}\frac{\partial \psi}{\partial r}$ will be made into equations 2 and 3, where ψ represents the stream function [7]. Then the relation

$$-\Omega = \frac{1}{r}\frac{\partial^2 \psi}{\partial r^2} - \frac{1}{r^2}\frac{\partial \psi}{\partial r} + \frac{1}{r}\frac{\partial^2 \psi}{\partial z^2} \tag{5}$$

will be applied, where Ω represents vorticity [7]. The substitution and (5) then leave us with the equation

$$\frac{\partial \Omega}{\partial t} + u\frac{\partial \Omega}{\partial r} + w\frac{\partial \Omega}{\partial z} = \frac{u\Omega}{r} + \eta\left(\frac{1}{r}\frac{\partial \Omega}{\partial r} - \frac{\Omega}{r^2} + \frac{\partial^2 \Omega}{\partial r^2} + \frac{\partial^2 \Omega}{\partial z^2}\right) \tag{6}$$

Note that by making the substitution $u = \frac{-1}{r}\frac{\partial \psi}{\partial z}$ and $w = \frac{1}{r}\frac{\partial \psi}{\partial r}$ and by applying (5), the momentum equations (2) and (3) transform into (6) which does not involve pressure [7]. Without making the substitutions for u and w, pressure would have to be solved for at each time-step which would severely increase computational costs. Now, only ψ and Ω need to be solved for at each time-step and equations (5) and (6) will be used to calculate those variables. In order to solve (4) at each time-step, the values for u and w will need to be calculated from the relation $u = \frac{-1}{r}\frac{\partial \psi}{\partial z}$ and $w = \frac{1}{r}\frac{\partial \psi}{\partial r}$. Also, in order to obtain (5) and (6), it had to be assumed that the fluid in the container was incompressible, $(\nabla \cdot (\mathbf{v}) = 0)$. Again, this is a valid assumption because the mixing of fuel vapors with air results in a small change in density.

3 Species Fraction Algorithm

The algorithm that was used to solve for Y was an upwind differencing scheme for advective-diffusive equations presented in [6]. This algorithm calls for backwards differencing to be applied to the advective terms and centered differencing to be applied to the diffusive terms. Thus, (4) becomes

$$\frac{Y^{n+1}(j,i) - Y^n(j,i)}{\Delta t} + u^n(j,i)\frac{Y^n(j,i) - Y^n(j,i-1)}{\Delta r} + w^n(j,i)\frac{Y^n(j,i) - Y^n(j-1,i)}{\Delta z} =$$
$$D\left(\frac{Y^n(j,i+1) - 2Y^n(j,i) + Y^n(j,i-1)}{\Delta r^2} + \frac{1}{(i-1)\Delta r}\frac{Y^n(j,i+1) - Y^n(j,i-1)}{2\Delta r}\right.$$
$$\left. + \frac{Y^n(j+1,i) - 2Y^n(j,i) + Y^n(j-1,i)}{\Delta z^2}\right) \tag{7}$$

Solving (7) for $Y^{n+1}(j,i)$ we see that

$$Y^{n+1}(j,i) = Y^n(j,i) - \Delta t\Bigg(u^n(j,i)\frac{Y^n(j,i)-Y^n(j,i-1)}{\Delta r} + w^n(j,i)\frac{Y^n(j,i)-Y^n(j-1,i)}{\Delta z}$$
$$- D\frac{1}{(i-1)\Delta r}\frac{Y^n(j,i+1)-Y^n(j,i-1)}{2\Delta r} - D\frac{Y^n(j,i+1)-2Y^n(j,i)+Y^n(j,i-1)}{\Delta r^2}$$
$$- D\frac{Y^n(j+1,i)-2Y^n(j,i)+Y^n(j-1,i)}{\Delta z^2}\Bigg) \tag{8}$$

To avoid unnecessary memory accesses and to minimize computation time, all of the terms in front of each matrix reference were gathered. This allows (8) to be written as

$$Y^{n+1}(j,i) = \left(1 - \frac{\Delta t}{\Delta r}u^n(j,i) - \frac{\Delta t}{\Delta z}w^n(j,i) - \frac{2D\Delta t}{\Delta r^2} - \frac{2D\Delta t}{\Delta z^2}\right)Y^n(j,i) + \frac{D\Delta t}{\Delta z^2}Y(j+1,i)$$
$$+ \left(\frac{\Delta t}{\Delta z}w^n(j,i) + \frac{D\Delta t}{\Delta z^2}\right)Y^n(j-1,i) + \left(\frac{D\Delta t}{\Delta r^2} + \frac{D\Delta t}{2(i-1)\Delta r^2}\right)Y^n(j,i+1)$$
$$+ \left(\frac{\Delta t}{\Delta r}u^n(j,i) + \frac{D\Delta t}{\Delta r^2} - \frac{D\Delta t}{2(i-1)\Delta r^2}\right)Y^n(j,i-1) \tag{9}$$

Thus, (9) allows the species fraction Y to be solved for at the next time-step and this is the equation that is used in the project code. At each time-step, all of the interior points of the domain are iterated through and afterwards the boundary conditions are implemented. This algorithm, is repeated until the final simulation time is reached.

4 Validation

Advective Diffusive Flow

To test the species fraction algorithm code, a case of pure oxygen being pumped into a cylinder of pure nitrogen was implemented. Instead of using the experimental domain in [1] and [2], a simpler domain of an axisymmetric cylindrical container was implemented. The boundary conditions were Neumann boundary conditions of $\frac{\partial Y}{\partial z} = 0$ on the top of the container and $\frac{\partial Y}{\partial r} = 0$ on the outer wall and inner axis of the container. On the bottom of the container, a Dirichlet boundary condition of $Y = 1$ was implemented. This meant that Y was tracking the O_2 fraction and that the initial condition would be $Y = 0$ to signify that the container was initially filled with pure N_2. Because pure oxygen is diffusing into nitrogen, $D = 0.25\frac{cm^2}{sec}$.

In this test case the radial velocity u was set to $0\frac{cm}{sec}$ and the axial velocity w was

set to a constant speed $c\frac{cm}{sec}$ over the entire domain. This implied that the outer wall of the cylinder was considered to have slip conditions. Several values of c were tested and Figure 2 presents the species fraction values for $c = 6\frac{cm}{sec}$ at various times.

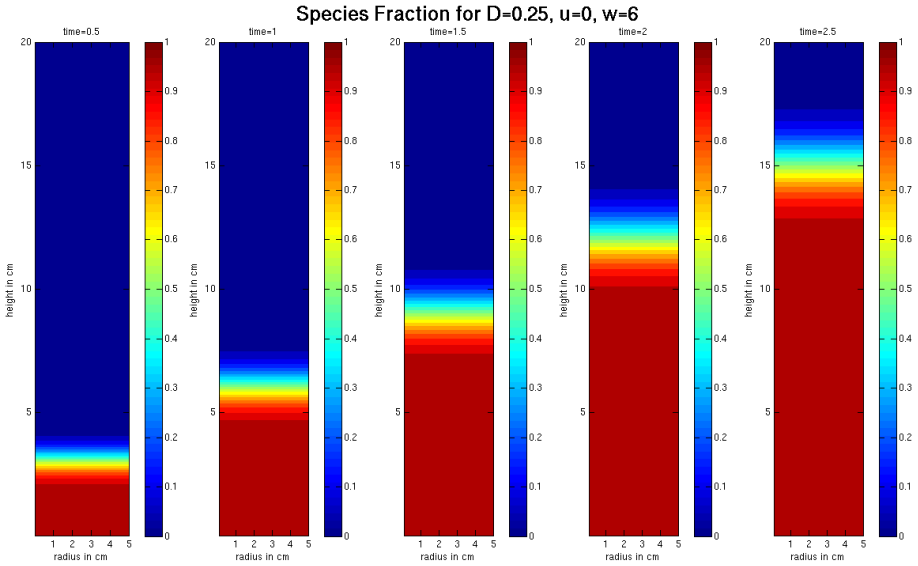

Figure 2: Species Fraction Field Solution for $D = 0.25\frac{cm^2}{sec}, u = 0\frac{cm}{sec}, w = 6\frac{cm}{sec}$ at times 0.5s to 2.5s in intervals of 0.5s. Values of 1.0 denote pure O_2 and values of 0.0 denote pure N_2.

Figure 2 shows that the diffusion layer's location is moving with time. Note that the layer appears to be found at height ct cm for all five times t. Parallel to expectations, the layer is appears to be moving correctly. Also parallel to expectations, the layer is growing with time due to the diffusion of O_2 into N_2 and vice versa. From these two visual aspects, it can be concluded to a high degree that the solver is working correctly.

Diffusive Flow

In order to further test the robustness of the species fraction code, two extreme cases were implemented. The first case was $D = 0.25\frac{cm^2}{sec}$, $u = 0\frac{cm}{sec}$, and $w = 10^{-5}\frac{cm}{sec}$ and it signifies a diffusion driven flow of O_2 into the container. The results of that test can be seen in Figure 3. Because the Δr value used in the code will have some effect on the solution, several values were tested and the corresponding species fraction field was calculated. (Because of how the code is set up, the Δz is calculated to be approximately the Δr value. Thus, as the Δr value changes so does the Δz value.) Figure 4 is a plot of the different Δr values and the respective species fraction values for $r = 2$ at different times. From Figure 4, it can been seen that the solution is similar between all four Δr

values. Assuming that the $\Delta r = 0.01cm$ solution is exact, Figure 5 is a plot of the error for the $\Delta r = 0.08cm$ solution and the $\Delta r = 0.02cm$ solution. Figure 5 shows the error drops by a factor of 10 when the Δr value drops by a factor of 4. This reflects that the diffusive terms, those that are the focus in this test case, are discretized by using centered differencing. When the grid spacing is dropped by a factor of 2, one expects the error to drop by a factor of 4 since the centered differencing is second order accurate.

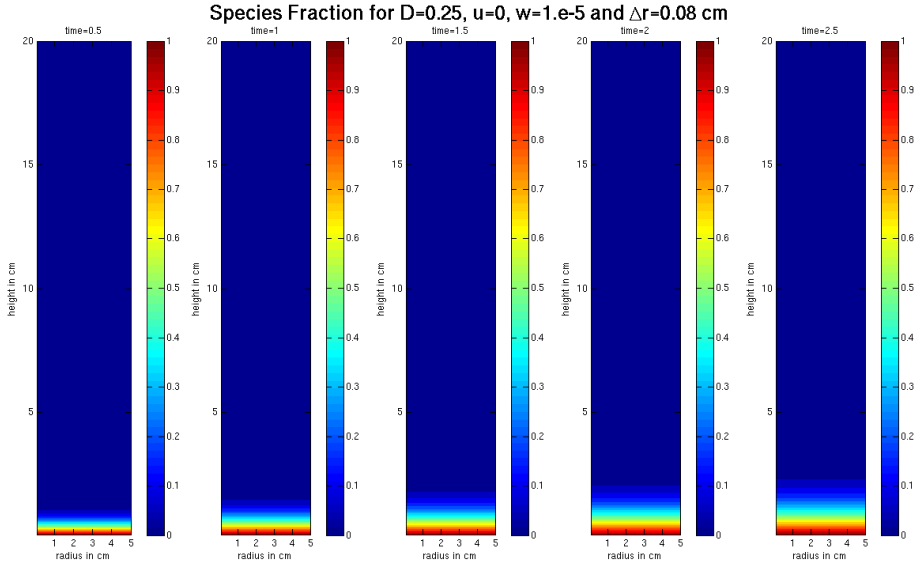

Figure 3: Species Fraction Field Solution for $D = 0.25\frac{cm^2}{sec}, u = 0\frac{cm}{sec}, w = 10^{-5}\frac{cm}{sec}$ at times 0.5s to 2.5s in intervals of 0.5s. Values of 1.0 denote pure O_2 and values of 0.0 denote pure N_2.

Advective Flow

The second test case was $D = 10^{-5}\frac{cm^2}{sec}$, $u = 0\frac{cm}{sec}$, $w = 5\frac{cm}{sec}$. These values signify an advective driven flow and this case will highlight how much numerical diffusion is taking place in the model. Figure 6 is a plot of this advective driven flow for $\Delta r = 0.08cm$. Looking at this graph, we can see that there appears to be a significant amount of diffusion taking place because the interface is not sharp. To investigate this further, several Δr were tested and these values were the same Δr values used in the previous case. Figure 7 is a plot of the different Δr values and their $Y(r = 2)$ solutions at several times. Looking at the results for $\Delta r = 0.08cm$ it is clear that the diffusive layer interface curve is not as sharp as it should be for $D = 10^{-5}\frac{cm^2}{sec}$. As Δr decreases, this interface becomes much sharper. Ultimately, at $\Delta r = 0.01cm$ numerical diffusion becomes very small and the interface

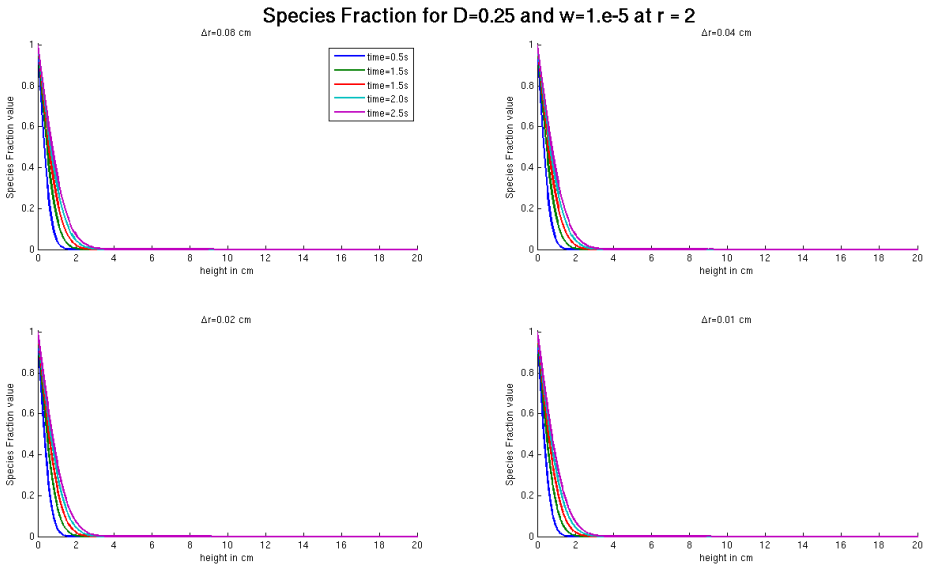

Figure 4: Species Fraction Solution at $r = 2$ for $D = 0.25\frac{cm^2}{sec}, u = 0\frac{cm}{sec}, w = 10^{-5}\frac{cm}{sec}$ at times 0.5s to 2.5s in intervals of 0.5s for multiple values of Δr. Values of 1.0 denote pure O_2 and values of 0.0 denote pure N_2.

becomes sharp. Thus, to minimize the effect of numerical diffusion in future simulations, a value of $\Delta r = 0.01cm$ or smaller will be used. Assuming the $\Delta r = 0.01cm$ is the correct solution, Figure 8 plots the error for the $\Delta r = 0.08cm$ solution and the $\Delta r = 0.02cm$ solution. Looking at Figure 8, we see that when the grid spacing is decreased by a factor of 4 the error decreases by a factor of 4 also. This agrees with the upwind differencing scheme because the advective terms, those that are driving the solution in this test case, are discretized according to a first order scheme. Thus, as the grid spacing decreases by a factor of 2, the error is expected to decrease by a factor of 2 also.

5 Stream function and Vorticity Algorithm

In order to solve equations (5) and (6), an algorithm presented in [6] will be used at each time-step. To begin, Ω is calculated from the current u and w fields. This is done following the relation that $\Omega = \frac{\partial u}{\partial z} - \frac{\partial w}{\partial r}$. This allows for the vorticity to be calculated according to the current u and w fields. The current vorticity field maybe be used instead of calculating Ω from u and w, but the u and w fields have updated after solving for Ω and ψ at the previous time-step. To maintain the highest degree of accuracy and to stay true to the algorithm in [6], the current values of u and w will be used to recalculate the

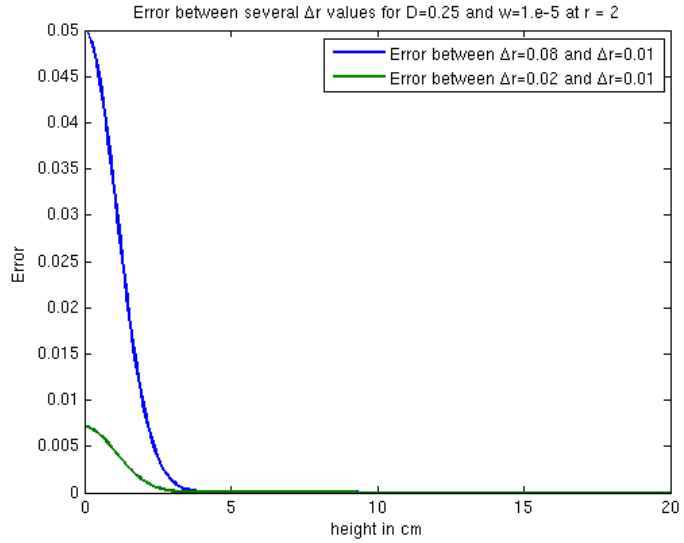

Figure 5: Error for solutions of Y between $\Delta r = 0.08cm$ and $\Delta r = 0.02cm$ and between $\Delta r = 0.08cm$ and $\Delta r = 0.01cm$ for $D = 0.25\frac{cm^2}{sec}$ and $w = 10^{-5}\frac{cm}{sec}$.

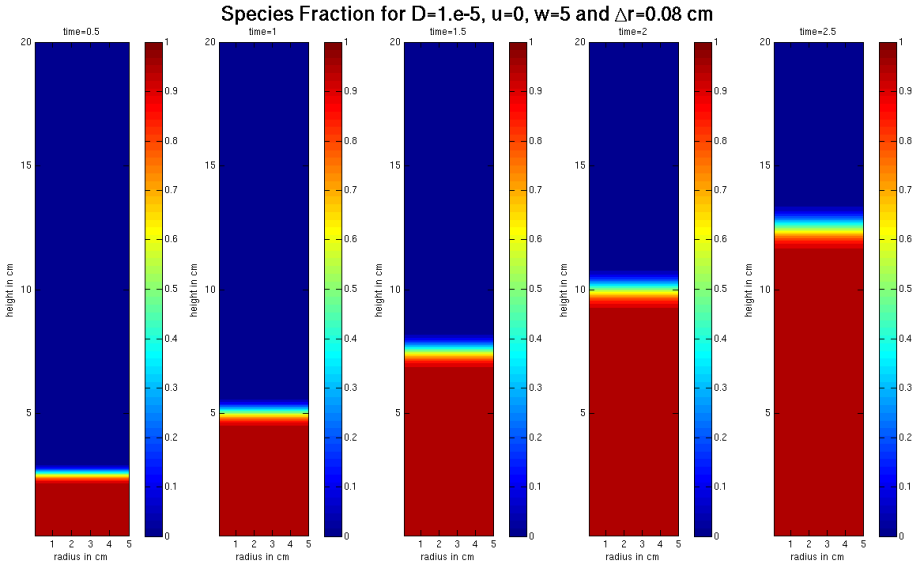

Figure 6: Species Fraction Field Solution for $D = 0.25\frac{cm^2}{sec}, u = 0\frac{cm}{sec}, w = 6\frac{cm}{sec}$ at times 0.5s to 2.5s in intervals of 0.5s. Values of 1.0 denote pure O_2 and values of 0.0 denote pure N_2.

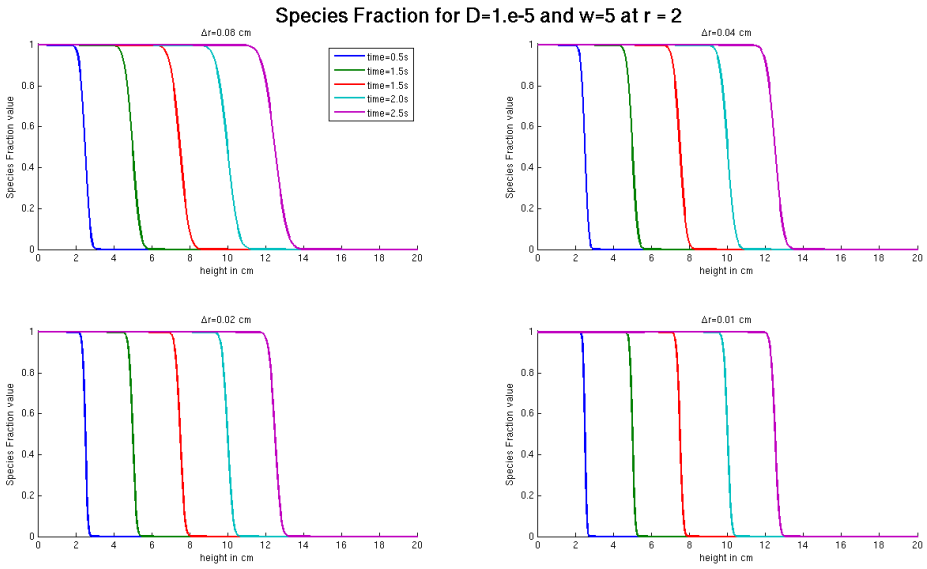

Figure 7: Species Fraction Solution at $r = 2$ for $D = 0.25\frac{cm^2}{sec}, u = 0\frac{cm}{sec}, w = 10^{-5}\frac{cm}{sec}$ at times 0.5s to 2.5s in intervals of 0.5s for multiple values of Δr. Values of 1.0 denote pure O_2 and values of 0.0 denote pure N_2.

Ω instead of using the current Ω values. This ensures that the Ω values are in complete agreeance with the current velocity values.

Once Ω^n has been calculated, Ω^{n+1} needs to be found. To do this, (6) will be used. Because (6) is an advective diffusive equation, the same upwind differencing algorithm used to solve (4) will be used to solve (6). Applying the upwind differencing discretization, (6) becomes

$$\frac{\Omega^{n+1}(j,i) - \Omega^n(j,i)}{\Delta t} + u^n(j,i)\frac{\Omega^n(j,i) - \Omega^n(j,i-1)}{\Delta r} + w^n(j,i)\frac{\Omega^n(j,i) - \Omega^n(j-1,i)}{\Delta z} =$$
$$\frac{\Omega^n(j,i)u^n(j,i)}{(i-1)\Delta r} + \eta\left(\frac{1}{(i-1)\Delta r}\frac{\Omega^n(j,i+1) - \Omega^n(j,i-1)}{2\Delta r} - \frac{\Omega^n(j,i)}{(i-1)^2\Delta r^2}\right.$$
$$\left. + \frac{\Omega^n(j,i+1) - 2\Omega^n(j,i) + \Omega^n(j,i-1)}{\Delta r^2} + \frac{\Omega^n(j+1,i) - 2\Omega^n(j,i) + \Omega^n(j-1,i)}{\Delta z^2}\right) \quad (10)$$

Solving (10) for $\Omega^{n+1}(j,i)$ we see that

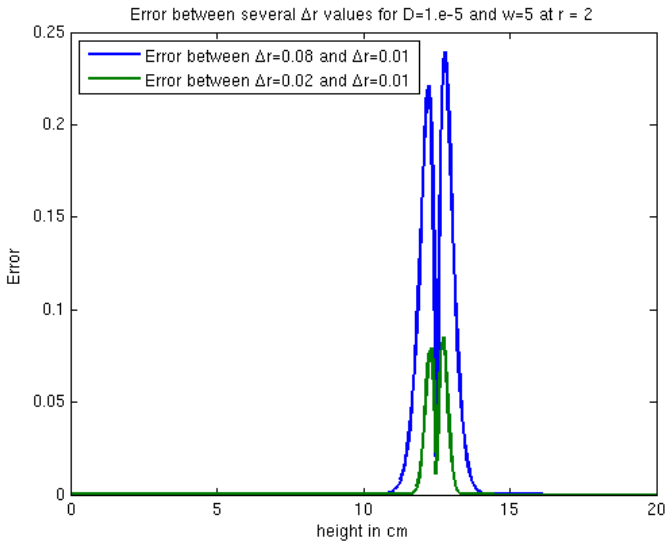

Figure 8: Error for solutions of Y between $\Delta r = 0.08cm$ and $\Delta r = 0.02cm$ and between $\Delta r = 0.08cm$ and $\Delta r = 0.01cm$ for $D = 0.25\frac{cm^2}{sec}$ and $w = 10^{-5}\frac{cm}{sec}$.

$$\begin{aligned}\Omega^{n+1}(j,i) =& \Omega^n(j,i) + \Delta t\Bigg(\frac{\Omega^n(j,i)u^n(j,i)}{(i-1)\Delta r} + \eta\frac{1}{(i-1)\Delta r}\frac{\Omega^n(j,i+1)-\Omega^n(j,i-1)}{2\Delta r} \\ &- \eta\frac{\Omega^n(j,i)}{(i-1)^2\Delta r^2} + \eta\frac{\Omega^n(j,i+1)-2\Omega^n(j,i)+\Omega^n(j,i-1)}{\Delta r^2} \\ &+ \eta\frac{\Omega^n(j+1,i)-2\Omega^n(j,i)+\Omega^n(j-1,i)}{\Delta z^2} - u^n(j,i)\frac{\Omega^n(j,i)-\Omega^n(j,i-1)}{\Delta r} \\ &- w^n(j,i)\frac{\Omega^n(j,i)-\Omega^n(j-1,i)}{\Delta z}\Bigg) \end{aligned} \tag{11}$$

Again, to avoid unnecessary memory accesses and to minimize computation time, all of the terms in front of each matrix reference were gathered. This allows (11) to be written as

$$\begin{aligned}\Omega^{n+1}(j,i) =&\left(1-\frac{\Delta t}{\Delta r}u^n(j,i)-\frac{\Delta t}{\Delta z}w^n(j,i)+\frac{\Delta t}{\Delta r(i-1)}u^n(j,i)-\frac{\Delta t\eta}{(i-1)^2\Delta r^2}-\frac{2\Delta t\eta}{\Delta r^2}-\frac{2\Delta t\eta}{\Delta z^2}\right)\\ &\Omega^n(j,i)+\left(\frac{\Delta t}{\Delta r}u^n(j,i)-\frac{\Delta t\eta}{2(i-1)\Delta r^2}+\frac{\Delta t\eta}{\Delta r^2}\right)\Omega^n(j,i-1)+\frac{\Delta t\eta}{\Delta z^2}\Omega^n(j+1,i)\\ &+\left(\frac{\Delta t\eta}{2(i-1)\Delta r^2}+\frac{\Delta t\eta}{\Delta r^2}\right)\Omega^n(j,i+1)+\left(\frac{\Delta t}{\Delta z}w^n(j,i)+\frac{\Delta t\eta}{\Delta z^2}\right)\Omega^n(j-1,i)\end{aligned} \tag{12}$$

Thus, (12) allows for Ω to be solved for at the next time-step and is the formula that is used in the project code.

The next step of the algorithm presented in [6], is to solve for ψ at the next time-step by using (5) and the recently calculated values of Ω^{n+1}. To solve (5), I have chosen to implement a Successive Over Relaxation (SOR) iterative algorithm that uses Chebyshev Acceleration to calculate the relaxation parameter ω. Before discussing the specifics of the SOR algorithm and Chebyshev Acceleration , let (5) be discretized. Upon discretization, it can be shown that

$$\begin{aligned}-\Omega^{n+1}(j,i) =&\frac{1}{(i-1)\Delta r}\left(\frac{\psi^{n+1}(j,i+1)-2\psi^{n+1}(j,i)+\psi^{n+1}(j,i-1)}{\Delta r^2}\right)\\ &-\frac{1}{(i-1)^2\Delta r^2}\left(\frac{\psi^{n+1}(j,i+1)-\psi^{n+1}(j,i-1)}{2\Delta r}\right)\\ &+\frac{1}{(i-1)\Delta r}\left(\frac{\psi^{n+1}(j+1,i)-2\psi^{n+1}(j,i)+\psi^{n+1}(j-1,i)}{\Delta z^2}\right)\end{aligned} \tag{13}$$

Moving the forcing term $\Omega^{n+1}(j,i)$ to the right hand side and calling the formula res, we have

$$\begin{aligned}res =&\Omega^{n+1}(j,i)+\frac{1}{(i-1)\Delta r}\left(\frac{\psi^{n+1}(j,i+1)-2\psi^{n+1}(j,i)+\psi^{n+1}(j,i-1)}{\Delta r^2}\right)\\ &-\frac{1}{(i-1)^2\Delta r^2}\left(\frac{\psi^{n+1}(j,i+1)-\psi^{n+1}(j,i-1)}{2\Delta r}\right)\\ &+\frac{1}{(i-1)\Delta r}\left(\frac{\psi^{n+1}(j+1,i)-2\psi^{n+1}(j,i)+\psi^{n+1}(j-1,i)}{\Delta z^2}\right)\end{aligned} \tag{14}$$

If the correct solution field ψ^{n+1} is known, then $res = 0$ by definition. However, if the solution field for ψ^{n+1} is not known and an incorrect solution is used, then $res \neq 0$. Thus, if ψ^n were put into (14) for all of the values of ψ^{n+1}, $res \neq 0$ because ψ^n is not in agreeance with the forcing term Ω^{n+1}.

The SOR algorithm calls for ψ^n to put into (14) and to be updated according to the formula

$$\psi(j,i) = \psi(j,i) - \omega\frac{res}{e} \tag{15}$$

where e is the constant term in front of $\psi^{n+1}(j,i)$ in the formula in (14) and ω is the relaxation parameter. The formula (15) is to be iterated multiple times until the value of the solution field's residual falls beneath a specified tolerance threshold. The method for measuring the residual of the solution fields is up to the programmer. For this model, the measurement will be the summation of $|res|$ for all of the interior points.

The SOR algorithm and its resulting formula, (15), is much faster than the Jacobi and Gauss-Seidel iterations and is guaranteed to converge as long as a $\omega \in (0,2)$. There is no way to foresee the exact number of iterations needed to reduce the residual of the solution field below the specified threshold. But there is a method to dynamically calculate the best ω value which, in turn dramatically reduces the number of iterations required. This method is Chebyshev Acceleration [8].

Chebyshev Acceleration calls for $\omega = 1$ for the first iteration through all of the interior points. This value of ω signifies that the first iteration is a Gauss-Seidel iteration. The second value of ω is then set to

$$\omega = \frac{1}{1 - 0.5q^2} \tag{16}$$

where

$$q = \frac{\cos(\frac{\pi}{m}) + \frac{\Delta r^2}{\Delta z^2}\cos(\frac{\pi}{n})}{1 + \frac{\Delta r^2}{\Delta z^2}} \tag{17}$$

After the first two iterations, the value for ω^k, the ω value for the k^{th} iteration, is determined by

$$\omega^k = \frac{1}{1 - 0.25q^2\omega^{k-1}} \tag{18}$$

A series of computational experiments were run on this algorithm. These experiments involved monitoring ω to ensure that $\omega \in (0, 2)$. By ensuring $\omega \in (0, 2)$, it is guaranteed that the SOR method will converge. For this system and the values of Δr and Δz that were tested, ω was always larger than 1 and never larger than 1.75. After multiple iterations, the ω value converged to approximately 1.64. Thus, we have a high amount of confidence that the condition that $\omega \in (0, 2)$ is satisfied. The optimal Ω value, 1.64, could be hard coded into the program to circumvent the Chebyshev Acceleration, but by changing ω according to the Chebyshev Acceleration faster convergence is guaranteed when compared to simply using the optimal ω value.

6 Boundary Conditions

During the early stages of coding, an effort has been made to create test cases that have the same initial and boundary conditions that the model of the experiments in [1] and [2] will have. This is true of the boundary conditions that were implemented in the testing of the species fraction model. Keeping with this effort and maintaining the simple axisymmetric cylindrical container that was used in the species fraction tests, a set of boundary conditions regarding u and w were devised. At the bottom of the container, $u = 0\frac{cm}{sec}$ and $w = c\frac{cm}{sec}$ where c is a specified inlet velocity. This signifies that the flow into the container is purely axial. On the outside wall of the container no slip conditions are implemented which implies that $u = w = 0\frac{cm}{sec}$. At the top of the container, there will be an outlet flow which means $\frac{\partial u}{\partial z} = \frac{\partial w}{\partial z} = 0$. Finally, on the interior axis, symmetry must be maintained so $\frac{\partial u}{\partial r} = \frac{\partial w}{\partial r} = 0$.

Since the problem has been changed from solving for u and w to ψ and Ω, boundary conditions for ψ and Ω need to be calculated from the boundary conditions for u and w. The boundary conditions that arise are as follows. At the bottom of the container $\Omega = 0$ and $\psi = 0.5c(r^2 - R^2)$ where R is the overall radius of the container (5cm). On the inner axis of the container, $\frac{\partial \psi}{\partial r} = \frac{\partial \Omega}{\partial r} = 0$. For the top of the container, $\frac{\partial \psi}{\partial z} = \frac{\partial \Omega}{\partial z} = 0$. Finally, on the outer wall of the container, $\psi = 0$ and $\frac{\partial \Omega}{\partial r} = 0$.

The boundary conditions for ψ and Ω, including those for Y, are boundary conditions for inlets, axes of symmetry, outlets, and no slip walls. These boundaries will be used to the model of the experiments in [1] and [2], which is why a large amount of time was spent designing test cases and deriving boundary conditions for the axisymmetric cylindrical container. By doing so, the boundary conditions required for the model of the experiments in [1] and [2] are already derived.

7 Future Work and Revised Timeline

Staying true to my original timeline, I will debug and validate my axisymmetric cylindrical container code during the month of December, and if necessary January. Origi-

nally, I hoped to have the Y, ψ, and Ω solvers programmed by this time along with having an analytical solution for stagnation flow. During my coding, I felt that it was more prudent to debug and test the Y solver code before coding the ψ and Ω solvers. By doing this, I could have confidence that any errors encountered while debugging the axisymmetric cylindrical container code could be found in the ψ and Ω solvers. Thus, in the respect of debugging, I am ahead of schedule. However, this came at the expense of pushing the stagnation flow solution back to a later date. I feel that this switch will save me valuable time in the spring semester.

In order to follow through with my original plan, I will be finding the analytical solution for stagnation flow in January. This will allow me to validate the results of the ψ and Ω solvers. To further validate the Y solver, I will also be working on finding the analytical solution of (4) for the case of $D = 0.25\frac{cm^2}{sec}$, $u = 0\frac{cm}{sec}$, and $w = 0\frac{cm}{sec}$ and for the case of $D = 0.0\frac{cm^2}{sec}$, $u = 0\frac{cm}{sec}$, and $w = 5\frac{cm}{sec}$. (This has been tested numerically and presented earlier in this paper.)

During the month of February, I will introduce the pipe and fritted ceramic disk from [1] and [2] into the cylindrical container. After creating these additions to the domain, I will test the code against the known diffusion constant for fuel vapors in air as was planned originally. The remaining months, March and April, will be spent wrapping up any loose ends and applying the code to the experimental datasets of [1] and [2] in order to find the diffusion constant of fuel vapors in film and in foam. The month of May, will be spent writing my final paper and presentation.

8 Platform and Deliverables

The model is being written in Fortran90 and compiled using the gfortran compiler. The simulations are being run on a MacBook Pro containing a 2.4 GHz Intel Core 2 Duo processor and 3 GB of memory. After the project is completed, a software package that is capable of finding diffusion coefficients for a fuel type in a film or foam will be delivered. The experimental concentration data that served as an input to the model will also be delivered.

9 References

1. Leonard, J.T., and Burnett, J.C. "Suppression of Fuel Evaporation by Aqueous Films of Fluorochemical Surfactant Solutions". NRL Report 7247. 1974

2. Williams, B.A., Murray, T., Butterworth, C., Sheinson, R.S., Fleming, J., Whitehurst, C., and Farley, F. "Extinguishment and Burnback Tests of Fluorinated and Fluorine-free Firefighting Foams with and without Film Formation." Suppression, Detection, and Signaling Research and Applications- A Technical Working Conference (SUPDET 2011). March 22-25, 2011. Orlando, Florida.

3. Williams, B.A, Sheinson, R.S., and Taylor, J.C. "Regimes of Fire Spread Across an AFFF – Covered Liquid Pool". NRL Report. 2010

4. Ananth, R, and Farley, J.P. "Suppression Dynamics of a Co-Flow Diffusion Flame with High Expansion Aqueous Foam". Journal of Fire Sciences. 2010

5. Bird, R.B., Stewart, W.E., and Lightfoot, E.N. *Transport Phenomena.* John Wiley and Sons. 1960

6. Pozrikidis, C. *Introduction to Theoretical and Computational Fluid Dynamics.* Oxford University Press. 1997

7. Panton, R.L. *Incompressible Flow.* John Wiley and Sons. 1984.

8. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P. *Numerical Recipes in Fortran.* Cambridge University Press. 1992