2013

DEVELOPMENT OF A 2-DIMENSIONAL FINITE VOLUME MODEL TO ASSESS HYDRODYNAMIC AND MICROBIAL CONTROLS ON DNAPL DISSOLUTION AND DETOXIFICATION

Eric S. Wesseldyke
Michigan Technological University

DEVELOPMENT OF A 2-DIMENSIONAL FINITE VOLUME MODEL TO ASSESS HYDRODYNAMIC AND MICROBIAL CONTROLS ON DNAPL DISSOLUTION AND DETOXIFICATION

By

Eric S. Wesseldyke

A THESIS

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

In Civil Engineering

MICHIGAN TECHNOLOGICAL UNIVERSITY

2013

This thesis has been approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE in Civil Engineering.

Department of Civil and Environmental Engineering

Thesis Co-Advisor:	*Eric A. Seagren*
Thesis Co-Advisor:	*Jennifer G. Becker*
Committee Member:	*Alex S. Mayer*
Committee Member:	*David R. Shonnard*
Department Chair:	*David W. Hand*

Table of Contents

Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 1034700. A portion of the research was performed using Environmental Molecular Sciences Laboratory (EMSL), a national scientific user facility sponsored by the Department of Energy's Office of Biological and Environmental Research and located at Pacific Northwest National Laboratory (PNNL). Dr. Changyong Zhang (PNNL, EMSL) assisted with the experimental work.

Preface

This thesis is original work produced by the author, Eric S. Wesseldyke, under the primary direction of Eric A. Seagren and Jennifer G. Becker. None of the text of the thesis is taken directly from previously published or collaborative articles. Portions of Chapter 4 form the basis of a manuscript that is in preparation. Eric S. Wesseldyke developed the mathematical model used in the study, performed the simulations that generated the modeling data and initial analyses of these data, and wrote the draft of Chapter 4. Eric A. Seagren and Jennifer G. Becker provided oversight of the model development and model simulations, contributed to the data analysis, and edited drafts of Chapter 4. Alex S. Mayer provided guidance regarding the numerical methods used in the model development and edited sections of the thesis that pertain to numerical methods. Eric A. Seagren and Jennifer G. Becker wrote the draft manuscript with input from Eric S. Wesseldyke and Alex S. Mayer.

Abstract

Tetrachloroethene (PCE) and trichloroethene (TCE) form dense non-aqueous phase liquids (DNAPLs), which are persistent groundwater contaminants. DNAPL dissolution can be "bioenhanced" via dissolved contaminant biodegradation at the DNAPL-water interface. This research hypothesized that: (1) competitive interactions between different dehalorespiring strains can significantly impact the bioenhancement effect, and extent of PCE dechlorination; and (2) hydrodynamics will affect the outcome of competition and the potential for bioenhancement and detoxification. A two-dimensional coupled flow-transport model was developed, with a DNAPL pool source and multiple microbial species. In the scenario presented, Dehalococcoides mccartyi 195 competes with Desulfuromonas michiganensis for the electron acceptors PCE and TCE. Simulations under biostimulation and low velocity (v_x) conditions suggest that the bioenhancement with Dsm. michiganensis alone was modestly increased by Dhc. mccartyi 195. However, the presence of Dhc. mccartyi 195 enhanced the extent of PCE transformation. Hydrodynamic conditions impacted the results by changing the dominant population under low and high v_x conditions.

1 Introduction

Contamination of groundwater with chlorinated ethenes such as tetrachloroethene (PCE) and trichloroethene (TCE) is common and frequently leads to the formation of dense non-aqueous phase liquids (DNAPLs). The U.S. EPA estimates that DNAPL contamination is likely present at approximately 60% of all National Priority List sites in the Superfund Program (1993). The DNAPL contaminant mass entrapped in saturated porous media can be present in a number of different geometries, which, due to the low aqueous solubility of PCE and TCE, can serve as a source of contamination for hundreds of years under natural conditions. Thus, the presence of a DNAPL source zone can greatly extend the time frames needed to reduce dissolved contaminant concentrations to regulatory levels using bioremediation.

Several approaches for remediating DNAPL source zones have been evaluated. The focus of this study is on stimulating the activity of the microbial population that biodegrades the aqueous-phase contaminant, thereby reducing its concentration at the DNAPL-water interface. This increases the driving force for contaminant dissolution from NAPL source zones relative to abiotic dissolution processes.

Engineered bioremediation approaches targeting aqueous-phase chlorinated ethenes generally are designed to alleviate any rate limitations on dehalorespiration—a form of anaerobic respiration in which the chlorinated ethene serves as the terminal electron acceptor and undergoes reductive dechlorination. This usually involves the implementation of biostimulation through the addition of an appropriate electron donor (often H_2) and/or bioaugmentation with cultures containing dehalorespiring populations. Two categories of dehalorespiring populations may be naturally present or added to contaminated groundwater via bioaugmentation (Figure 1.1). The first category encompasses several dehalorespiring bacteria, including *Desulfuromonas michiganensis*, that dehalorespire PCE and TCE and produce *cis*-dichloroethene (DCE) as the dominant dechlorination product. Complete detoxification of chlorinated ethenes to ethene can be achieved only if sufficient numbers of dehalorespirers in the second category—*Dehalococcoides mccartyi* strains—are present, because these organisms appear to be unique in their ability to dehalorespire DCE and/or vinyl chloride (VC).

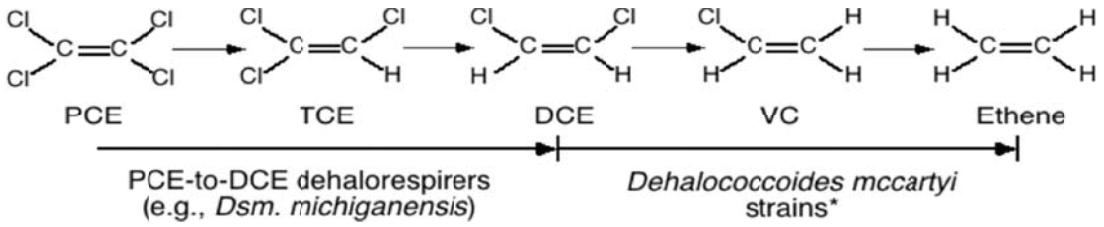

Figure 1.1. Sequential reductive dechlorination of PCE to non-toxic ethene mediated by two categories of dehalorespiring populations: PCE-to-DCE respiring populations such as *Dsm. michiganensis* and *Dhc. mccartyi* strains that are able to respire DCE and/or VC. Note that some *Dehalococcoides* (*Dhc.*) strains can respire higher chlorinated ethenes, and *Dhc. mccartyi* 195 can completely dechlorinate PCE to ethene, although it transforms VC cometabolically.

Modeling studies (Becker and Seagren 2009; Christ et al. 2005; Seagren et al. 1993; Seagren et al. 1994) have theoretically demonstrated the potential for biodegradation-enhanced NAPL dissolution or "bioenhanced dissolution" to decrease the longevity of NAPL source zones. This potential has been verified in laboratory studies (Carr et al. 2000; Cope and Hughes 2001; Seagren et al. 2002; Yang and McCarty 2000; Yang and McCarty 2002), and evidence of bioenhanced dissolution has also been obtained in field studies (Aulenta et al. 2007; Essaid et al. 2003; Sorenson 2003). While the results of these studies are promising, the extent to which bioenhanced dissolution can be applied to remediate DNAPL contaminant source zones in the field is still poorly understood (AFCEE 2004). Little is known about the factors actually controlling biological reaction kinetics in situ because the laboratory studies conducted to date have primarily been proof-of-concept experiments that utilized one-dimensional (1-D) columns and focused on demonstrating the occurrence of bioenhanced dissolution without attempting to optimize this phenomenon (Carr et al. 2000; Cope and Hughes 2001; Seagren et al. 2002; Yang and McCarty 2000; Yang and McCarty 2002). The modeling studies that have been conducted to date provide more detailed analyses; however, the insight into the factors controlling bioenhanced dissolution provided by these studies is limited by their utilization of 1-D mathematical models and/or failure to incorporate key phenomena that may impact bioenhanced dissolution in real groundwater systems.

In particular, different dehalorespiring bacteria may be present within the microbial community inhabiting a DNAPL source zone and complex competitive or complementary ecological interactions may occur between these and other populations (Becker, 2006). These ecological interactions have generally been ignored in the models developed to examine bioenhancement phenomena. However, Becker and Seagren (2009) used a 1-D, cells-in-series model to show that these ecological interactions play a critical role in determining which dehalorespiring population becomes most abundant at the DNAPL-water interface. Their work also showed that hydrodynamic factors, especially the relative rates of advection and chlorinated ethene biodegradation, also affect the outcome of these ecological interactions. Importantly, because of significant differences in the dehalorespiration kinetics exhibited by different dehalorespiring strains, the extent to which bioenhanced dissolution was observed depended largely on the dehalorespiring population that was dominant at the DNAPL-water interface.

Although this earlier modeling study highlights the importance of developing mathematical models that incorporate the ecological interactions within the microbial community at the DNAPL source zone and the interplay between biological and hydrodynamic processes, several important hydrodynamic phenomena and aspects of the flow environment cannot be described using a 1-D model. For example, the rate of advection along the interface between the aqueous phase and the NAPL strongly influences the dissolution rate along the NAPL boundary that would be present without bioenhancement (Seagren and Moore 2003). The advection rate may also determine the most appropriate method for modeling the rate of contaminant dissolution from the NAPL to the aqueous phase, i.e., a local equilibrium or mass-transfer-limited model. In addition to the advection rate in the initial flow field, the accumulation of microbial

biomass within pore spaces can influence hydrodynamic conditions and therefore the rate of contaminant dissolution. The effects of this "bioclogging" on hydrodynamics and dissolution rates cannot be evaluated using 1-D models; however, a 2-D modeling study of bioenhanced dissolution of DNAPLs showed that bioclogging can be significant when biostimulation is implemented (Chu et al. 2003).

The current study was undertaken to build on the advances made in these earlier studies and overcome several shortcomings in our understanding of bioenhanced dissolution of DNAPLs. Specifically, the overall goal of this research was to develop and validate a refined 2-D model that incorporates the ecological interactions among dehalorespiring populations, as well as the non-linear equations used to describe dehalorespiration kinetics, and can be used to assess the key hydrodynamic, kinetic, and ecological phenomena that affect contaminant dissolution from the NAPL to the aqueous phase. Without this information, the true potential for remediating DNAPL contaminant source zones using bioenhanced dissolution cannot be accurately assessed.

The work reported here is part of a larger National Science Foundation-funded study that includes experimental evaluation of bioenhanced dissolution by defined co-cultures of dehalorespiring bacteria in a microfluidic groundwater model (micromodel) and intermediate-scale flow cell (ISFC) or sand tank reactor. The overall hypotheses governing this study are:

(1) Different chlorinated ethane-respiring strains may inhabit and dominate different regions within DNAPL source zones and dissolved contaminant plumes due to the ecological interactions among dehalorespiring populations for chlorinated ethenes and/or electron donors;

(2) The outcome of ecological interactions can significantly impact the degree of bioenhancement of the dissolution rate and, thus, DNAPL source zone longevity, as well as the extent of chlorinated ethane dechlorination; and

(3) Hydrodynamics will affect the outcome of the ecological interactions and the potential for bioenhancement and detoxification.

These hypotheses were investigated using an integrated numerical modeling and experimental approach. Specifically, the objectives of this study were to:

(1) develop a conceptual model of a 2-D domain with DNAPL dissolution and transport, and competition between multiple microbial species in co-culture being tracked individually, while growing and performing reductive dechlorination;

(2) develop a working mathematical model in Matlab to represent the conceptual model developed;

(3) use the mathematical model to help design a micromodel experimental system;

(4) experimentally determine mass transfer and mass transport model parameters; and

(5) use the mathematical model and parameter estimates to systematically simulate DNAPL bioenhancement, aqueous contaminant dechlorination, and microbial distribution in the micromodel reactor for a key scenario with different combinations of flow rate, substrate supply, and microbial community composition.

Ultimately, the Matlab model will also be used to design the micromodel and ISFC experiments that will be used to test the model predictions.

The remainder of this thesis comprises the following chapters: Chapter 2, Model Development, explains the development of both the conceptual and mathematical models to achieve Objectives (1) and (2). It also documents the validation of predictions made using the numerical model developed in Matlab through comparison to the solutions of appropriate analytical models. Chapter 3, Model Implementation, serves as a "user manual" for future users of the Matlab model described in Chapter 2. Chapter 4 presents the experimental and modeling methods used to achieve Objectives (3) and (4) and analysis of the results obtained during the systematic simulation of DNAPL bioenhancement, aqueous contaminant dechlorination, and microbial distribution in the micromodel reactor for one example scenario, as per Objective (5). Chapter 4 also provides a brief summary of the conclusions that can be drawn from those results.

2 Model Development

2.1 Goal of the Model

As reviewed in the previous chapter, numerous one-dimensional column studies have been performed to estimate the bioenhancement of dense non-aqueous phase liquid (DNAPL) dissolution under engineered biostimulation conditions. However, there have been relatively few multi-dimensional experimental, e.g., (Seagren et al. 2002), and numerical modeling, e.g., (Chu et al. 2003), investigations of bioenhanced NAPL dissolution. Furthermore, the impact of microbial competition on bioenhanced dissolution has only been evaluated in a simple one-dimensional domain (Becker and Seagren 2009; Christ et al. 2005; Seagren et al. 1993; Seagren et al. 1994). The overall goal of this work is to develop a model that fills this knowledge gap, by incorporating two-dimensional flow and contaminant transport in a porous medium that includes a DNAPL source, and an adaptable biological reaction component that simulates microbial competition.

The model is used to simulate dissolution from a DNAPL pool of tetrachloroethene (PCE). The model couples dissolution of the DNAPL source with the simultaneous transport and fate of dissolved PCE and two exogenous electron donors, acetate and dissolved hydrogen. The fate of the PCE is controlled by reductive dechlorination; therefore, the model includes the transport and fate of the daughter products of PCE reductive dechlorination as well (TCE, DCE, VC, and ethene). The model also incorporates that activity and growth of up to four microbial species, which use some combination of the chemicals listed above as electron donors and acceptors.

One further key consideration in the model development was that the model needed to be able to describe the processes of interest in the experimental systems used in the laboratory component of this work and the larger project of which it was a part. Accordingly, the model was initially designed to describe the processes of interest in a micromodel flow cell. However, the numerical model also needed to have the versatility to be scaled up to an intermediate-scale flow cell system (on the order of meters in scale). Given the initial focus on the micromodel scale, a pore-scale modeling approach such as the Lattice-Boltzmann Finite Volume Method (LBFVM) could have been implemented, e.g., (Willingham et al. 2010; Willingham et al. 2008). However, it has been demonstrated in other studies (Knutson et al. 2007) that the pore-scale model results can be appropriately predicted with a continuum model if the dispersivity values are fitted properly. With these considerations in mind, a continuum-based model domain was chosen to allow scale-up to a larger system.

Finally, the numerical model also needed to be designed with a built-in mass balance calculation for the chemical species of concern. This was important for checking the accuracy of the numerical solution, and for comparison with the laboratory data collected in other components of this project. Specifically, in the laboratory systems, the only way to estimate the dissolution rate and bioenhancement effect will be by using a mass

balance on the chlorinated ethenes exiting the system. Thus, to facilitate a mass balance, based on the known boundary conditions, the model calculates the rate of dissolution of PCE at each time step, as well as the rate of efflux of each of the dissolved chemical species.

2.2 Solute Transport Governing Equation

The transport of a soluble contaminant in a two-dimensional porous medium is described by the advective-dispersive-reactive (ADR) transport equation shown below.

$$\frac{\partial S}{\partial t} = \frac{\partial}{\partial x}\left(D_x \frac{\partial S}{\partial x} - v_x S \right) + \frac{\partial}{\partial y}\left(D_y \frac{\partial S}{\partial y} - v_y S \right) \pm \left(\frac{\partial S}{\partial t} \right)_{Rxn} \tag{2.1}$$

Where: S = substrate concentration [M/L^3], t = time [T], x = distance from the inlet [L], y = distance from the NAPL interface [L], D_{ij} = hydrodynamic dispersion in the x- or y-direction [L^2/T], v_{ij} = the average pore-water velocity in the x- or y-direction [L/T], and $\left(\partial S/\partial t\right)_{Rxn}$ is a reaction term. For this model, the reaction term applies to the biokinetics of either production or degradation of the solute. The x-direction is the direction parallel to the DNAPL-water interface, which is also the primary flow direction with $x = 0$ located at the beginning of the DNAPL pool. The y-direction is transverse to that, with the DNAPL-water interface located at $y = 0$.

The hydrodynamic dispersion term consists of both molecular diffusion and hydrodynamic dispersivity, and is calculated as follows:

$$D_i = \frac{\alpha_l v_i^2 + \alpha_t v_j^2}{\sqrt{v_i^2 + v_j^2}} + \tau d_i \tag{2.2}$$

where: $\alpha_{l,t}$ = the longitudinal and transverse dispersivity, respectively, [L]; τ = the tortuosity factor [dimensionless]; and d_i = molecular diffusion coefficient, assumed to be equal in the x- or y-direction, respectively [L^2/T].

2.3 Boundary Conditions

Solution of the transport equations for the chemical species of interest in a two-dimensional system requires specification of four boundary conditions. For the model presented in this thesis, the boundary conditions were selected to describe the micromodel experimental system, although they could be used to describe several other systems. The four distinct boundaries of the rectangular domain corresponding to the

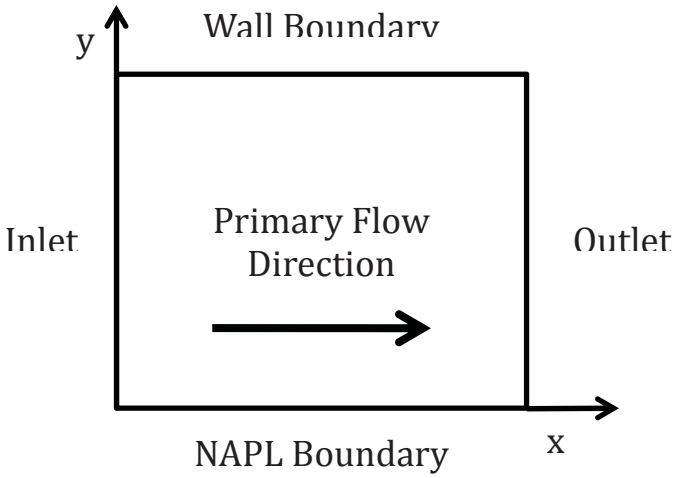

Figure 2.1. The four distinct boundaries of the rectangular domain corresponding to the micromodel reactor system.

micromodel reactor system are illustrated in Figure 2.1. Henceforth, these boundaries are referred to as the inlet (left), the outlet (right), the DNAPL side (bottom), and the wall side (top).

To represent the inlet side, which will be under the influence of a constantly applied flow, a constant flux boundary condition was selected. This simulates the effect of a pump, injecting media at a constant rate with constituents (e.g., electron donor(s)) at a known concentration. Accordingly, the Darcy flux (q_{x0}) and the concentration are held constant at each point.

At the outlet side of the model, a free outlet is simulated. Such a boundary allows mass to freely exit the domain via advection and dispersion (Versteeg and Malalasekera 2007). The specific details of implementation of this boundary condition are described in a later section.

The wall side of the model is simulated as a zero-flux boundary. Accordingly, there is no chemical transport through that side of the model. Implementation of this boundary condition is described in a later section of this chapter.

The DNAPL boundary of the model is simulated as two possible dissolution conditions for PCE and a zero-flux boundary for the other chemical (TCE, DCE, etc.). The first condition is that of local equilibrium (LE), and the second is mass transfer limited (MTL), or nonequilibrium condition (Seagren et al. 1999). Under the LE assumption, the PCE is assumed to dissolve into the domain at a rate sufficiently fast relative to competing PCE solute sinks (i.e., advection, dispersion, and biodegradation) that the aqueous concentration of PCE at the interface of the aqueous-phase and NAPL is equal to the solubility of PCE. Under this assumption, transverse dispersion and diffusion control the movement of PCE into the system.

If the rates of removal of PCE solute in the system via advective-dispersive and reactive sinks are sufficiently greater in magnitude than the PCE interphase mass transfer rate, MTL conditions will occur. Under the MTL condition, it is possible for the aqueous concentration of PCE at the water-NAPL interface to fall below solubility. If this is the case, then the interphase mass transfer rate of PCE from the DNAPL into the aqueous phase will control the flux of PCE into the system and, thus, the mass flux away from the pool via transverse dispersion. The specific details of the numerical implementation of the LE and MTL boundary conditions for PCE are given in Section 2.6.

2.4 Biodegradation Kinetics

In order to produce biomass and energy, microorganisms need three substrates: an electron donor, an electron acceptor (either external or internal), and a carbon source (Nester 2009). The Monod equations, shown below, have been commonly used to relate the utilization rate of the limiting substrate to the biomass and substrate concentration:

$$\frac{dS}{dt} = -\hat{q}\left(\frac{S}{S+K}\right)X \tag{2.3}$$

$$\frac{dX}{dt} = \left[Y\hat{q}\left(\frac{S}{S+K}\right) - b\right]X \tag{2.4}$$

where: S represents the concentration of the substrate [M/L^3], X represents the concentration of the biomass [M_X/L^3], $\hat{q}$ represents the maximum substrate utilization rate [M/T], K is the concentration of the substrate in which the rate of utilization will be one-half of its maximum [M/L^3], Y is the true yield [M_X/M_S], and b is the decay rate coefficient [T^{-1}].

Under circumstances where more than one substrate may limit the rate of utilization, and the subsequent growth, the Monod equation as seen above will not suffice. One modification is to make a system of equations using dual-Monod kinetics. The general system is shown below:

$$\frac{dS}{dt} = \hat{q}\left(\frac{S}{S+K_S}\right)\left(\frac{A}{A+K_A}\right)X \tag{2.5}$$

$$\frac{dA}{dt} = f\hat{q}\left(\frac{S}{S+K_S}\right)\left(\frac{A}{A+K_A}\right)X \tag{2.6}$$

$$\frac{dX}{dt} = \left[Y\hat{q}\left(\frac{S}{S + K_S} \right)\left(\frac{A}{A + K_A} \right) - b \right] X \tag{2.7}$$

The new terms introduced are *A* and *f*. In this formulation, *S* represents the concentration of the electron donor [M/L^3], and *A* represents the concentration of the electron acceptor [M/L^3] . The new variable *f* is a fraction which relates the number of moles of *S* that will be needed to react with one mole of *A* (M_A/M_S). As a consequence, it is also necessary to choose by which substrate $\hat{q}$ will be defined.

The model developed for this project uses a combination of single- and dual-Monod kinetics. Single-Monod kinetics are used for the methanogenic population. For the dehalorespiring populations, a set of dual-Monod equations defines the interaction of the biomass with each of the electron donor and acceptor pairs that it can utilize, while also tracking products. A further explanation of the details of these equations is given in the next chapter.

2.5 Operator Splitting

As seen above, the equation describing the reactive transport of each chemical is made up of two components: a linear partial differential equation (PDE) describing advective and dispersive transport and a non-linear ordinary differential equation describing reactions. It is also in the reaction term where the equation for each contaminant will depend on the concentration of the other contaminants. Taking all of this into account, a coupled system of over seven non-linear PDEs needs to be solved. Solution of non-linear equations can require a great deal of computational effort, and in a two-dimensional system, this effort would lead to computational times that are far greater than desired.

One method that has been devised to address this issue is called Operator Splitting (OS). As the name suggests, OS splits the solution of the equations into two parts, with the advective-dispersive (AD) PDEs solved separately from the reaction ODEs. It has been demonstrated that operator splitting will converge on an accurate solution, with an inherent mass-balance error that can be minimized by reducing the time step (Carrayrou et al. 2004; Valocchi and Malmstead 1992).

There are several methods of arranging the split of the two equations, but all non-iterative methods follow the same basic procedure. First, either the AD equation or the reaction system of equations is solved for a time step. The concentrations at each location produced by this solution are then used as the input for the other equation and solved over the same time step. The method of splitting called Strang OS was selected for this model, as it has been shown to be second-order accurate on global mass balance with a constant first-order reaction. In Strang OS, the AD equation is solved for the first half of the time step. The concentrations of the solutes at each location are used as the initial values of the system of ODEs for biological reaction, which is solved over the full time step. The

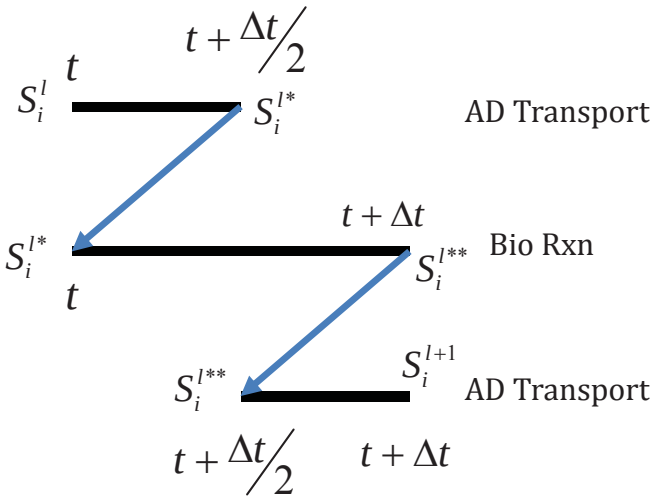

Figure 2.2. Sequence of calculations with the Strang OS solution technique.

concentrations from the biological reaction are then substituted as the initial values of AD transport to be solved for the second half of the time step. This process is illustrated in Figure 2.2.

Using OS, the simulations will take less computational effort, because the linear PDE for the AD part of the equation can be solved using linear algebra and the coupling and non-linearity of the biological reactions can be solved separately.

2.6 Finite Volume Method (Non-Reactive AD)

The AD part of the governing equation is solved by a method known as the Finite Volume Method (FVM). FVM was selected for this model because it is relatively simple to implement and locally conservative, unlike Finite Differences (FD) or Finite Elements (FE) (Versteeg and Malalasekera 2007).

Application of the FVM to the governing equation for this system is demonstrated as follows. The starting point is the isolated AD PDE,

$$\frac{\partial S}{\partial t} = \frac{\partial}{\partial x}\left(D_x \frac{\partial S}{\partial x} - v_x S \right) + \frac{\partial}{\partial y}\left(D_y \frac{\partial S}{\partial y} - v_y S \right) \tag{2.8}$$

and the control volume (CV). Each CV is rectangular in this formulation, with the layout illustrated in Figure 2.3.

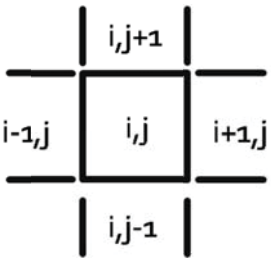

Figure 2.3. Layout of rectangular control volumes (CV).

Next, the governing equation is integrated over the domain of the example CV, $\Omega_{i,j}$.

$$\iint_{\Omega_{i,j}} \left[\frac{\partial S}{\partial t} \right] dxdy = \iint_{\Omega_{i,j}} \left[\frac{\partial}{\partial x}\left(D_x \frac{\partial S}{\partial x} - v_x S \right) + \frac{\partial}{\partial y}\left(D_y \frac{\partial S}{\partial y} - v_y S \right) \right] dxdy \tag{2.9}$$

$$\left[\frac{\partial S}{\partial t} \right] \Delta x_{i,j} \Delta y_{i,j} = \Delta y_{i,j} \left[\left(D_x \frac{\partial S}{\partial x} - v_x S \right) \right]_{i-\frac{1}{2},j}^{i+\frac{1}{2},j} + \Delta x_{i,j} \left[\left(D_y \frac{\partial S}{\partial y} - v_y S \right) \right]_{i,j-\frac{1}{2}}^{i,j+\frac{1}{2}} \tag{2.10}$$

Note that the resulting terms on the right-hand side of the equation quantify the flux of solute through each of the four sides of the rectangle.

Applying a first-order finite difference approximation for the time derivative and using the Crank-Nicholson method for the evaluation of the right-hand-side of the equation, Eq. (2.10) can be rewritten as follows.

$$\begin{aligned} \frac{\left(S_{i,j}^{l+1} - S_{i,j}^{l} \right)}{\Delta t} \Delta x_i \Delta y_j = \theta & \left[\Delta y_j \left(D_x \frac{\partial S}{\partial x} - v_x S \right) \Big|_{i-\frac{1}{2},j}^{i+\frac{1}{2},j} + \Delta x_i \left(D_y \frac{\partial S}{\partial y} - v_y S \right) \Big|_{i,j-\frac{1}{2}}^{i,j+\frac{1}{2}} \right]^{l+1} \\ & + (1-\theta) \left[\Delta y_j \left(D_x \frac{\partial S}{\partial x} - v_x S \right) \Big|_{i-\frac{1}{2},j}^{i+\frac{1}{2},j} + \Delta x_i \left(D_y \frac{\partial S}{\partial y} - v_y S \right) \Big|_{i,j-\frac{1}{2}}^{i,j+\frac{1}{2}} \right]^{l} \end{aligned} \tag{2.11}$$

In eqn. 2.11 *l* is the time index. For this work a value of $\theta = 0.5$ was implemented, in which case the time derivative is second-order accurate.

The coefficients for the linear matrix are derived as follows. First, the term RHS is substituted in the right-hand side of the equation to represent the portion of the equation that encompasses the previous time step:

$$\frac{\left(S_{i,j}^{l+1}-S_{i,j}^{l}\right)}{\Delta t}\Delta x_i \Delta y_j = \theta\left[\Delta y_j\left(D_x\frac{\partial S}{\partial x}-v_x S\right)\Big|_{i-\frac{1}{2},j}^{i+\frac{1}{2},j}+\Delta x_i\left(D_y\frac{\partial S}{\partial y}-v_y S\right)\Big|_{i,j-\frac{1}{2}}^{i,j+\frac{1}{2}}\right]^{l+1}+\left(1-\theta\right)RHS$$

(2.12)

Next, by multiplying through by Δt and dividing by Δx_i and Δy_i the following expression is derived.

$$\begin{aligned}S_{i,j}^{l+1}-S_{i,j}^{l}=\theta\Bigg[&\frac{\Delta t}{\Delta x_i}\left(D_x\frac{\partial S}{\partial x}\Big|_{i+\frac{1}{2},j}-v_x S\Big|_{i+\frac{1}{2},j}\right)-\frac{\Delta t}{\Delta x_i}\left(D_x\frac{\partial S}{\partial x}\Big|_{i-\frac{1}{2},j}-v_x S\Big|_{i-\frac{1}{2},j}\right)\\&+\frac{\Delta t}{\Delta y_j}\left(D_y\frac{\partial S}{\partial y}\Big|_{i,j+\frac{1}{2}}-v_y S\Big|_{i,j+\frac{1}{2}}\right)-\frac{\Delta t}{\Delta y_j}\left(D_y\frac{\partial S}{\partial y}\Big|_{i,j-\frac{1}{2}}-v_y S\Big|_{i,j-\frac{1}{2}}\right)\Bigg]+\left(1-\theta\right)RHS\end{aligned}$$

(2.13)

At each internal interface, the concentration gradient is calculated using a centered finite difference and the concentration is estimated by a centered differencing scheme as follows, using a linear interpolation to account for the variable grid size:

$$\frac{\partial S}{\partial x}\Big|_{i+\frac{1}{2},j}\approx\frac{2\left(S_{i+1,j}-S_{i,j}\right)}{\Delta x_{i+1,j}+\Delta x_{i,j}}$$

(2.14)

$$S_{i+\frac{1}{2},j}\approx\frac{S_{i+1,j}\Delta x_{i,j}+S_{i,j}\Delta x_{i+1,j}}{\Delta x_{i+1,j}+\Delta x_{i,j}}$$

(2.15)

$$\frac{\partial S}{\partial y}\Big|_{i.\,j+\frac{1}{2}}\approx\frac{2\left(S_{i,j+1}-S_{i,j}\right)}{\Delta y_{i,j+1}+\Delta y_{i,j}}$$

(2.16)

$$S_{i,j+\frac{1}{2}}\approx\frac{S_{i,j+1}\Delta y_{i,j}+S_{i,j}\Delta y_{i,j+1}}{\Delta y_{i,j+1}+\Delta y_{i,j}}$$

(2.17)

By substituting the terms in Eq. (2.14) – (2.17) into Eq. (2.13) and rearranging the equation, the following coefficients can be derived for the internal CVs in the coefficient matrix.

$$S_{i,j}^{l+1}:1-\theta\left[-\frac{\Delta t}{\Delta x_i}\left(\frac{2D_{x,i+\frac{1}{2},j}}{\left(\Delta x_i+\Delta x_{i+1}\right)}+\frac{2D_{x,i-\frac{1}{2},j}}{\left(\Delta x_i+\Delta x_{i-1}\right)}+\frac{\Delta x_{i+1}v_{x,i+\frac{1}{2},j}}{\Delta x_i+\Delta x_{i+1}}-\frac{\Delta x_{i-1}v_{x,i+\frac{1}{2},j}}{\Delta x_i+\Delta x_{i-1}}\right)\right.$$
$$\left.-\frac{\Delta t}{\Delta y_j}\left(\frac{2D_{y,i,j+\frac{1}{2}}}{\left(\Delta y_j+\Delta y_{j+1}\right)}+\frac{2D_{x,i,j-\frac{1}{2}}}{\left(\Delta y_j+\Delta y_{j-1}\right)}+\frac{\Delta y_{j+1}v_{y,i,j+\frac{1}{2}}}{\Delta y_j+\Delta y_{j+1}}-\frac{\Delta y_{j-1}v_{y,i,j-\frac{1}{2}}}{\Delta y_j+\Delta y_{j-1}}\right)\right] \tag{2.18}$$

$$S_{i+1,j}^{l+1}:-\theta\left[\frac{\Delta t}{\Delta x_i}\left(\frac{2D_{x,i+\frac{1}{2},j}}{\Delta x_i+\Delta x_{i+1}}-\frac{\Delta x_i v_{x,i+\frac{1}{2},j}}{\Delta x_i+\Delta x_{i+1}}\right)\right] \tag{2.19}$$

$$S_{i-1,j}^{l+1}:-\theta\left[\frac{\Delta t}{\Delta x_i}\left(\frac{2D_{x,i-\frac{1}{2},j}}{\Delta x_i+\Delta x_{i-1}}+\frac{\Delta x_i v_{x,i-\frac{1}{2},j}}{\Delta x_i+\Delta x_{i-1}}\right)\right] \tag{2.20}$$

$$S_{i,j+1}^{l+1}:-\theta\left[\frac{\Delta t}{\Delta y_j}\left(\frac{2D_{y,i,j+\frac{1}{2}}}{\Delta y_j+\Delta y_{j+1}}-\frac{\Delta y_j v_{y,i,j+\frac{1}{2}}}{\Delta y_j+\Delta y_{j+1}}\right)\right] \tag{2.21}$$

$$S_{i,j-1}^{l+1}:-\theta\left[\frac{\Delta t}{\Delta y_j}\left(\frac{2D_{y,i,j-\frac{1}{2}}}{\Delta y_j+\Delta y_{j-1}}+\frac{\Delta y_i v_{y,i,j-\frac{1}{2}}}{\Delta y_j+\Delta y_{j-1}}\right)\right] \tag{2.22}$$

The right-hand side (RHS) vector is calculated in a similar fashion as described above. Therefore, substituting the RHS into Eq. (2.13) and placing the terms on their respective side of the equation results in the following approximation:

$$S_{i,j}^{l+1}-\theta\left[\frac{\Delta t}{\Delta x_i}\left(D_x\frac{\partial S}{\partial x}\bigg|_{i+\frac{1}{2},j}-v_x S\big|_{i+\frac{1}{2},j}\right)-\frac{\Delta t}{\Delta x_i}\left(D_x\frac{\partial S}{\partial x}\bigg|_{i-\frac{1}{2},j}-v_x S\big|_{i-\frac{1}{2},j}\right)\right.$$
$$\left.+\frac{\Delta t}{\Delta y_j}\left(D_y\frac{\partial S}{\partial y}\bigg|_{i,j+\frac{1}{2}}-v_y S\big|_{i,j+\frac{1}{2}}\right)-\frac{\Delta t}{\Delta y_j}\left(D_y\frac{\partial S}{\partial y}\bigg|_{i,j-\frac{1}{2}}-v_y S\big|_{i,j-\frac{1}{2}}\right)\right]^{l+1}$$
$$=S_{i,j}^{l}+(1-\theta)\left[\frac{\Delta t}{\Delta x_i}\left(D_x\frac{\partial S}{\partial x}\bigg|_{i+\frac{1}{2},j}-v_x S\big|_{i+\frac{1}{2},j}\right)-\frac{\Delta t}{\Delta x_i}\left(D_x\frac{\partial S}{\partial x}\bigg|_{i-\frac{1}{2},j}-v_x S\big|_{i-\frac{1}{2},j}\right)\right.$$
$$\left.+\frac{\Delta t}{\Delta y_j}\left(D_y\frac{\partial S}{\partial y}\bigg|_{i,j+\frac{1}{2}}-v_y S\big|_{i,j+\frac{1}{2}}\right)-\frac{\Delta t}{\Delta y_j}\left(D_y\frac{\partial S}{\partial y}\bigg|_{i,j-\frac{1}{2}}-v_y S\big|_{i,j-\frac{1}{2}}\right)\right]^{l} \tag{2.23}$$

The substitutions listed in Eqs. (2.14) to (2.17) are applied to Eq. (2.23) to calculate the RHS vector for each time step.

The model boundary conditions are each applied by modifying Eq. 2.8 appropriately. For the fixed-flux boundary condition, the flux term $\left(D_x \left.\frac{\partial S}{\partial x}\right|_{i-\frac{1}{2},j} - v_x S\big|_{i-\frac{1}{2},j} \right)$ in both the future and current time steps is replaced by $v_x S_0$, where S_0 is the influent concentration of the solute. At the no flux boundary conditions, such as at y_L and at $y = 0$ for every solute except PCE, the flux terms representing the no-flux boundaries are set equal to zero. The free outlet is simulated by adding a column of control volumes outside of the domain, and setting the hydrodynamic dispersion for the outlet of the extra volumes equal to zero. Therefore, there is still the possibility of dispersion flux through the location where the efflux is calculated. The Matlab code for this process is included in Appendix A.

2.7 Flow Field Calculations

As illustrated in Equation 2.8, the equation for chemical transport requires the pore-water velocity and the hydrodynamic dispersion coefficient, which is a function of the pore-water velocity, as inputs. Based on Darcy's Law, the average pore water velocity is proportional to the hydraulic gradient, with the proportionality factor, K, termed as the hydraulic conductivity [L/T]:

$$v_x = -\frac{K}{n}\left(\frac{dh}{dx}\right) \tag{2.24}$$

The variable n represents the porosity of the media through which water passes. This is the ratio of the volume of voids to the total volume of the sample (unitless).

Preliminary simulation results by the author in a similar modeling system revealed the possibility for growth of large quantities of biomass in the pore network, which could effect a change in the local hydraulic conductivity and porosity. A previous modeling study by Chu et al. (2003) demonstrated that such biomass-induced changes in hydraulic conductivity can have a significant effect on the level of bioenhancement measured. The authors simulated the change in hydraulic conductivity using two different assumptions: (1) the biomass grew as a uniform biofilm, and (2) the biomass formed aggregate plugs in the pore throats in which it was present. The same approach was taken in this research.

Biomass accumulation also affects the porosity. Therefore, the model needs a method for adjusting the biomass growth rates to both allow for change with time but to also not overload the pore spaces. As discussed in Section 2.4, the substrate utilization rate for

each species of biomass is calculated using Monod or dual-Monod kinetics. After this has been calculated, the following equations describe the growth rates of each of two microbial populations in competition, using *Dhc. mccartyi 195* and *Dsm. michiganensis* BB1 as an example,

$$\frac{dX_{195}}{dt} = Y_{195}\left(r_{ut,195,PCE} + r_{ut,195,TCE} + r_{ut,195,DCE}\right) - b_{195}X_{195} \tag{2.25}$$

$$\frac{dX_{BB1}}{dt} = Y_{BB1}\left(r_{ut,BB1,PCE} + r_{ut,BB1,TCE}\right) - b_{BB1}X_{BB1} \tag{2.26}$$

The fractions of pore space filled by biomass of each population are calculated as follows (Chu et al. 2003):

$$n_{b,195} = \frac{nX_{195}}{X_{f,195}} \tag{2.27}$$

$$n_{b,BB1} = \frac{nX_{BB1}}{X_{f,BB1}} \tag{2.28}$$

Where X_f is the biofilm density (M_X/L_X^3) of each species, assumed to be constant at 8000 mg VSS/L for each species in this study. When the total pore volume occupied by both populations is the total pore volume, the following equation is true.

$$n = n_{b,195} + n_{b,BB1} = \frac{nX_{195}}{X_{f,195}} + \frac{nX_{BB1}}{X_{f,BB1}} \tag{2.29}$$

$$\frac{X_{195}}{X_{f,195}} + \frac{X_{195}}{X_{f,195}} = 1 \tag{2.30}$$

Thus, Equation (2.30) is used as the condition to trigger the response to stop the total biomass from increasing further. Although the total amount of biomass cannot change, the fraction of the pore space occupied by each population can change as a result of competition. Therefore, the following restriction is placed on the biomass growth:

$$\frac{dX_{195}}{dt} + \frac{dX_{BB1}}{dt} = 1 \tag{2.31}$$

Conceptually, this is implemented by calculating the amount of decay that would occur, then keeping the growth rates in the same ratio to each other to fill the space back in. Therefore, if the rate equations (2.25) and (2.26) are substituted into Eq. (2.31), the following equation can be developed.

$$\begin{aligned}\omega Y_{195}\left(r_{ut,195,PCE}+r_{ut,195,TCE}+r_{ut,195,DCE}\right)-b_{195}X_{195}\\ +\omega Y_{BB1}\left(r_{ut,BB1,PCE}+r_{ut,BB1,TCE}\right)-b_{BB1}X_{BB1}=0\end{aligned} \tag{2.32}$$

$$\begin{aligned}\omega\left[Y_{195}\left(r_{ut,195,PCE}+r_{ut,195,TCE}+r_{ut,195,DCE}\right)+Y_{BB1}\left(r_{ut,BB1,PCE}+r_{ut,BB1,TCE}\right)\right]\\ =b_{195}X_{195}+b_{BB1}X_{BB1}\end{aligned} \tag{2.33}$$

In eqns. 2.32 and 2.33 ω is a scaling factor for the growth rates to keep the net growth of total biomass equal to zero. Eq. (2.34) can be solved for ω:

$$\omega=\frac{b_{195}X_{195}+b_{BB1}X_{BB1}}{Y_{195}\left(r_{ut,195,PCE}+r_{ut,195,TCE}+r_{ut,195,DCE}\right)+Y_{BB1}\left(r_{ut,BB1,PCE}+r_{ut,BB1,TCE}\right)} \tag{2.34}$$

The scaling factor can then be applied to the biomass growth rate. The calculated substrate utilization rates are left unaffected, and the growth equations then become:

$$\frac{dX_{195}}{dt}=\omega\left[Y_{195}\left(r_{ut,195,PCE}+r_{ut,195,TCE}+r_{ut,195,DCE}\right)\right]-b_{195}X_{195} \tag{2.35}$$

$$\frac{dX_{BB1}}{dt}=\omega\left[Y_{BB1}\left(r_{ut,BB1,PCE}+r_{ut,BB1,TCE}\right)\right]-b_{BB1}X_{BB1} \tag{2.36}$$

To avoid numerical instability in the code, the growth equations are changed to this form when the pore volume is 99% full.

To implement Darcy's law in the model domain, the 2-D flow profile was assumed to reach equilibrium quickly compared to the transport phenomena. Correspondingly, the following steady state equation was used to describe the hydraulic head at each location in the domain.

$$0=\frac{\partial}{\partial x}\left(K_x\frac{\partial h}{\partial x}\right)+\frac{\partial}{\partial y}\left(K_y\frac{\partial h}{\partial y}\right) \tag{2.37}$$

Where h is the hydraulic head at the specific location [L], and x and y are the location in the x- or y-direction [L].

Much like the transport equation (Eq. 2.8), this is a linear PDE that can be solved using FVM. Assuming an isotropic system and using the same FVM technique, Eq. 2.11 can be integrated over the domain of a generic CV as follows:

$$\iint_{\Omega_{i,j}} 0 dxdy = \iint_{\Omega_{i,j}} \left[\frac{\partial}{\partial x}\left(K \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y}\left(K \frac{\partial h}{\partial y} \right) \right] dxdy \tag{2.38}$$

$$0 = \Delta y_{i,j} \left(K \frac{\partial h}{\partial x} \right)_{i,j-\frac{1}{2}}^{i,j+\frac{1}{2}} + \Delta x_{i,j} \left(K \frac{\partial h}{\partial y} \right)_{i-\frac{1}{2},j}^{i+\frac{1}{2},j} \tag{2.39}$$

Note from the above derivation that the Darcy flux is calculated on each edge of the control volume. The gradients are calculated at each face of the CV using a central finite difference approach, as was described above for the transport equation.

The porous medium is assumed to be isotropic; therefore $K = K_x = K_y$. However, the biomass does not grow uniformly, which means that K will not be uniform throughout the domain. Accordingly, the value of K is calculated by employing a harmonic average, because this is the accepted method for estimating the average conductivity transverse to the layering in an aquifer. For example,

$$\frac{\Delta x_i + \Delta x_{i+1}}{K_{i+\frac{1}{2},j}} = \frac{\Delta x_i}{K_i} + \frac{\Delta x_{i+1}}{K_{i+1}} \tag{2.40}$$

Therefore,

$$K_{i+\frac{1}{2},j} = \frac{\Delta x_i + \Delta x_{i+1}}{\left(\frac{\Delta x_i}{K_i} + \frac{\Delta x_{i+1}}{K_{i+1}} \right)} = \frac{\Delta x_i + \Delta x_{i+1}}{\left(\frac{K_{i+1}\Delta x_i + K_i \Delta x_{i+1}}{K_i K_{i+1}} \right)} \tag{2.41}$$

and

$$K_{i+\frac{1}{2},j} = \frac{\left(\Delta x_i + \Delta x_{i+1} \right) K_i K_{i+1}}{\left(K_{i+1}\Delta x_i + K_i \Delta x_{i+1} \right)} \tag{2.42}$$

The same principle is applied to calculate *K* for the other three sides of the control volume.

Note that in Eq. (2.39) the Darcy flux is calculated at each of the four faces of the CV. This allows for relatively simple implementation of the boundary conditions. At the no-flow boundaries, which are the NAPL side and the opposite wall side, the Darcy flux is set equal to zero. At the inlet side, the Darcy flux is set to a predetermined value based on the input values. At the outlet, a constant head boundary condition is set in a column of CVs outside of the domain. From the known distribution of the hydraulic head and the porosity, the pore water velocity is then calculated for each interface using the following relationship:

$$v_x = \frac{q_x}{n} \tag{2.43}$$

Where q_x is the Darcy flux and n is the porosity at that location. The porosity at the interface is the mean of the effective porosity in the adjacent CVs.

2.8 Solving Non-Linear ODEs

With the Strang OS technique, the AD portion of the governing ADR equation is solved for the first half of the time, step with the resulting solute concentrations used as the initial values for the non-linear reaction term. As described above, a strictly macroscopic Monod kinetics model of biomass growth and substrate utilization was selected. This method of quantification of growth and substrate utilization provides a pair of non-linear ODEs to be solved simultaneously for a single substrate (Eqs. 2.3 and 2.4), and a triplet of ODEs to be solved (Eq. 2.4-2.6) in the case of the reactions controlled by dual-substrate limitation kinetics, e.g. the dechlorination reactions. The situation is further complicated, however, by the interactions of multiple microbial species in competition and the sequential nature of the reductive dechlorinator reactions. For example, the rate of change of the TCE concentration is dependent on the PCE utilization of more than one microbial species, producing TCE, and the rate of TCE utilization by those same species. Thus, the complexity of the system ranges from two coupled, non-linear ODEs up to eleven.

There are several methods of solving non-linear ODEs numerically, each with advantages and disadvantages (Chapra and Canale 2010). For this work, the Cash-Karp formulation of the Runge-Kutta method (CK-RK) was selected. This is a single step method with an adaptive step size. CK-RK is a fourth-order RK method with a fifth-order corrector. For each calculation , the method requires only six evaluations, rather than a step-halving method of a similar order that would require ten or more, thereby saving on computational effort. The adaptive step size is necessary to handle the significant

changes in time that are possible for the solute concentrations, especially when electron donor concentrations get near the threshold concentration, below which the reaction will stop. To create a modular model code, which can be more easily amended by later users, the biological calculations are controlled in two parts. The first part is an overall controlling function to enact the CK-RK with a variable time step. The second part is an inner function that calculates the derivative side of the ODEs in vectorized format. This method was selected to mimic the method used in the Matlab ODE solvers, but using a different formulation and leaving out several computationally expensive features. The internal function can be easily written to describe any system of ODEs for the scenario of intent, provided the equations are kept in the correct order. In this way, future work applying more elaborate biofilm modeling or other kinetic assumptions can be performed using the same overall model framework.

Additional information regarding the implementation of CK-RK is available in the literature on the subject (Chapra and Canale 2010).

3 Model implementation

3.1 Platform and Recommended User Experience

The model development was performed in a Matlab environment. Although Matlab does not compile and run as quickly as lower-level languages, it has several built-in functions that make the development of an engineering model easier. The program is especially well suited toward solving linear systems of equations such as the advection-dispersion transport equations. Matlab also has a number of built-in ODE solvers with variable step-size and options appropriate for many situations; however, as will be evaluated further, these solvers are computationally "expensive" in the current system.

Although the main computational effort of the model is performed in Matlab, the user inputs are provided in a Microsoft Excel spreadsheet. This was done for two reasons. One, it allows easier visualization of the inputs as they are entered, and will make the model easier for users who are less Matlab-savvy. Two, the Matlab and Excel programs have also been designed to interface well and easily.

To facilitate use of this model, it is recommended that the user have an entry-level understanding of Matlab and Excel. Basic use should not require changing the code of the Matlab functions; however, more advanced tests or adaptation to additional scenarios would require making several changes, specifically writing new lines of code, or writing a new function for the biological reactions.

3.2 Program Design/Flow Chart

The flow of the overall program is summarized in the flow chart shown in Figure 3.1. When the master script for the simulation is run, it first loads all of the necessary workspace variables from a .mat file named "Initialization.mat" and sets up the run. The main loop begins after that, and it contains several parts. First, the coefficient matrices are formed, factored, and stored in memory. This action is only done when the flow network needs to be solved due to changes in hydraulic conductivity. Next, the program solves the advection-dispersion (AD) equations for the first half of the time step using Matlab's cgs solver. The concentrations of solutes are then substituted into the biological reactions, which are solved at each location independently for the length of the entire time step. Next, the AD equations are solved for the second half of the time step. Finally, the dechlorination rates are calculated at each location for each microbial species based on the concentrations at the end of the time step. These are recorded to vectors in memory.

All of the code for the simulation is included in Appendix A.

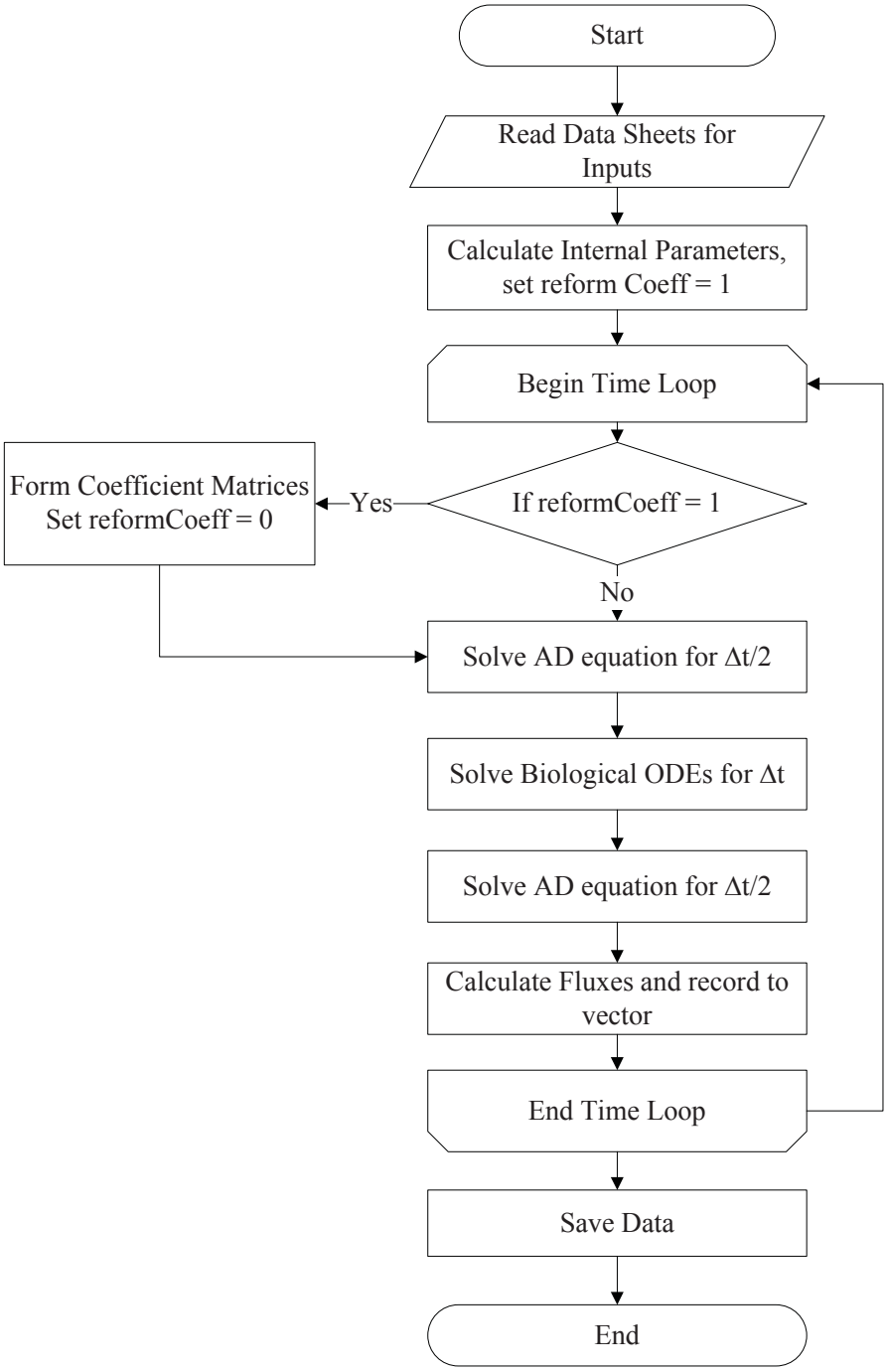

Figure 3.1. Flow Chart for the Overall Program

3.3 User Interaction: Inputs

The code was intentionally made flexible so that the domain may be scaled up, and to allow simulation of a range of relevant situations. As a result, there are several variables that the user must input to specify the simulation conditions. To make the data input more user-friendly, three Excel spreadsheets are used. In this way, the inputs are spread out, with the more commonly varied parameters grouped together, while keeping the others in a separate location to avoid accidentally changing them. Examples of each of the spreadsheets are included in Appendix B.

In the first spreadsheet, the most commonly changed parameters are edited. These include the dimensions of the domain (xL, yL, thickness), the porosity (n), the volumetric flow rate (Q), the longitudinal and transverse dispersivities (α_L, α_T), tortuosity (τ), hydraulic conductivity (K), hydrodynamic potential at the outlet (h_L), and the DNAPL dissolution mass-transfer coefficient. This spreadsheet is also where decisions are entered for the biological clogging modeling method, NAPL dissolution method, biological species present, and the time-related decisions (i.e. duration, time step, and report interval). Finally, the molecular diffusion coefficients in water are listed for each chemical that is tracked in the model. The second spreadsheet contains a discretization file, in which the user can select what the distribution of Δx and Δy values will be for the grid domain. The user is responsible to make sure that the total values of Δx and Δy add up to the correct lengths. The third spreadsheet of inputs contains the biological modeling parameters for the microbial species that the model can handle.

3.4 User Interaction: Outputs

The model saves the workspace variables to the hard drive of the computer at the time interval specified by the user in the input sheets. This allows the user to get a snapshot of the concentrations at all locations in the domain at the end of the specified time step. Also, given that mass balance is one of the primary methods of analysis for the experimental setup, the efflux rate of each chemical being tracked is calculated each time the AD equations are solved and saved in a matrix. The dissolution flux rate of PCE is calculated each time the AD equations are solved and saved in a matrix. In order to use the data collected, the user simply loads the saved workspace .mat file and then can freely manipulate it. This method of recording data is less automatic than if the model were to calculate each output parameter itself, but it is faster and more versatile. Also, the user can write a script to perform the same type of analysis several times in a row.

3.5 User Instructions

In order to run a simulation and generate data, there are three steps. First, the conditions must be defined in the Excel spreadsheets mentioned in Section 3.3. Second, the initialization file must be generated from the information entered, done by a pre-

determined function. Third, the simulation itself is run. The following is a more detailed description of these three stages.

The spreadsheets are opened using the function *editinputs* which was written by the author. Once all three spreadsheets are open, the user has the option to define each of the inputs.

ScenarioInputs.xlsx contains the most commonly changed inputs. It is listed first in Appendix B.

DiscretizationInputs.xlsx is used to define the discretization of the grid for the FVM calculations. In the column labeled deltaX each entry is the width of a column in the grid. The total of this column is also used as the length of the domain, and is read automatically. The first 10 columns are not in contact with the DNAPL source, and the rest are. Similarly, the column labeled deltaY is the width of each row of control volumes in the domain, with the first entry being closest to the DNAPL source. This spreadsheet is listed second in Appendix B.

BioInputs.xlsx is used to define the kinetic parameters for the species that can be simulated, using the units specified. When all three spreadsheets have been edited they must be saved and closed for the initialization step.

After the inputs have been defined, the simulation must be initialized. This reads the data from the spreadsheets and saves it to a Matlab data file (.mat) that contains all the necessary information to run the simulation. This is defined using the function *initializesimulation*. The file, named *Initialization.mat*, is saved in the folder *Model_Inputs.*

The simulation is run using *simulation_master,* which loads *Initialization.mat* into memory and begins doing the calculations. A list of how to make common, specific modifications to the code is provided in Appendix C as a quick reference guide.

4 Modeling Results and Discussion

4.1 Modeling Methods

Domain: Model evaluation of the effects of microbial competition on DNAPL pool dissolution was performed using a two-dimensional saturated porous medium, which is initially homogeneous and isotropic with the primary flow direction along the x-axis. Flow is two-dimensional, thereby allowing for the investigation of the effects of bioclogging, which has been shown to be important in other modeling studies of bioenhanced DNAPL dissolution (Chu et al. 2003). The domain is modeled with a third-type boundary condition at the inlet, a free outlet boundary condition, and a no-flow/no-flux boundary along the top of the domain. The boundary condition at the PCE DNAPL pool/aqueous phase interface along the bottom of the domain is modeled using either a local equilibrium (first-type) boundary condition, or a non-equilibrium (mass-transfer limited) boundary condition. Implementation of the boundary condition at the DNAPL pool source is described in more detail in (Chapter 2). All simulations reported in this manuscript were performed using the local equilibrium boundary condition, based on preliminary abiotic PCE dissolution studies that suggested this assumption was appropriate, as well as previous work with NAPL pools (Seagren et al. 1999).

Conceptual Models of Competition: The competition scenario modeled in this study involved *Dhc. mccartyi* 195 in competition with *Dsm. michiganensis* for electron acceptors PCE and TCE (Figure 4.1). However, the organisms are not in competition for the electron donor because *Dhc. mccartyi* 195 is restricted to the use of H_2 and *Dsm. michiganensis* is unable to utilize H_2 as an electron donor, but can grow on acetate and other organic compounds. This competition scenario is relevant to field conditions because both H_2 and acetate are produced during fermentation of lactate and other organic substrates that are commonly supplied to a contaminant plume during engineered bioremediation (e.g., (Lendvay et al. 2003).

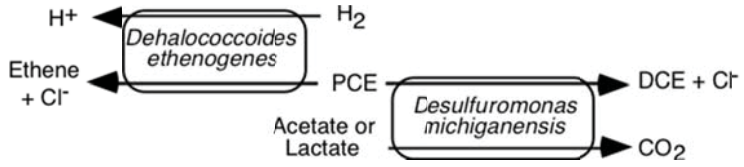

Figure 4.1. Scenario 1 conceptual model of competition. Note that prior to 2013, *Dhc. mccartyi* 195 was named *Dhc. ethenogenes* (strain 195). Therefore, the name *Dhc. ethenogenes* appears in this and other figures throughout the thesis.

Numerical Model: Previous numerical modeling studies, e.g., (Knutson et al. 2007; Willingham et al. 2010), indicate good agreement between pore- and 2-D continuum-scale models, if the dispersivity values are fitted properly. Based on this, and the need to later up-scale the model to a larger experimental system, a continuum-scale modeling approach was selected. Therefore, reactive solute transport through the domain was

modeled using the non-steady-state form of the 2-D advection-dispersion-reaction (ADR) equation, shown here for the electron acceptor substrate:

$$\frac{\partial A_i}{\partial t} = \frac{\partial}{\partial x}\left(D_x \frac{\partial A_i}{\partial x} - v_x A_i \right) + \frac{\partial}{\partial y}\left(D_y \frac{\partial A_i}{\partial y} - v_y A_i \right) - q_{\max,i} X_i \left(\frac{S_i}{K_{S,i} + S_i} \right)\left(\frac{A_i}{K_{A,i} + A_i} \right) \qquad (4.1)$$

where A_i is the solute i of interest, D_x and D_y are the hydrodynamic dispersion coefficient in the x- and y-directions, v_x and v_y are the pore-water velocity in the x- and y-directions, $q_{max,i}$ is the maximum specific substrate utilization rate, and $K_{S,i}$ and $K_{A,i}$ are the half-rate concentrations of the electron donor, S_i, and the electron acceptor, A_i, respectively. Following the approach of Becker (2006) and Becker and Seagren (2009), the reaction terms for reductive dehalogenation are assumed to follow the dual-Monod kinetic model given a specific microbial species and electron donor solute of interest. Possible toxicity and inhibition effects have been neglected for this study. The above equation is coupled with an electron donor equation of the same form and one for biomass growth (See Chapter 2).

Eq. 4.1 can be transformed to a nondimensional form by defining $t^* = (t/(L_x v_x))$, $x^* = x/L_x$, $z^* = z/L_x$, $A^* = A_i/A_{eq,i}$, a longitudinal Peclet number, Pe_l, a transverse Peclet number, Pe_t, and Damköhler No. 2, Da_2, where L_x = NAPL pool length in the x-direction [L], $A_{eq,i}$ = aqueous solubility of compound i, and Pe_l, Pe_t, and Da_2 are defined in Table 4.1:

$$\frac{\partial A^*}{\partial t^*} = \frac{1}{Pe_l}\frac{\partial^2 A^*}{\partial x^{*2}} + \frac{1}{Pe_t}\frac{\partial^2 A^*}{\partial z^{*2}} - \frac{\partial A^*}{\partial x^*} - Da_2 X^* \left(\frac{S^*}{K_{e\text{-}\,donor}{}^* + S^*} \right)\left(\frac{A^*}{K_{A,i}{}^* + A^*} \right) \qquad (4.2)$$

To reduce the computational effort, Strang operator splitting (OS) was applied to separate the biological reaction system of non-linear ordinary differential equations (ODEs) from the linear partial differential equation (PDE) describing advection and dispersion (AD). Strang OS has been shown to be second-order accurate on global mass balance under specific circumstances, unlike other methods which were first-order accurate (Valocchi and Malmstead 1992). The 2-D AD PDE for each solute is solved for each time step using a 2-D finite volume method (FVM) with rectangular control volumes (CVs) and applying the Crank-Nicholson (CN) method for the time derivative (Versteeg and Malalasekera 2007). FVM is locally conservative, making it an appropriate choice when modeling flow with fast reactions, and CN is second-order accurate in time. Solution of the 2-D PDE for AD requires as inputs the average pore water velocity and hydrodynamic dispersion coefficient, which is a function of the average pore water velocity. To obtain these values, flow through the model domain was described using Darcy's Law. The two-dimensional flow profile was assumed to reach steady state quickly compared to the transport phenomena, and the following equation was used to describe the hydraulic head at each location in the domain:

$$0 = \frac{\partial}{\partial x}\left(K \frac{\partial h}{\partial x} \right) + \frac{\partial}{\partial y}\left(K \frac{\partial h}{\partial y} \right) \tag{4.3}$$

Table 4.1. Definition and significance of the dimensionless numbers

	Symbol	Definition	Significance with respect to rate-limiting process	
			Dimensionless number < 1	**Dimensionless number > 1**
Dimensionless numbers comparing mass transfer rates	Pe (Peclet No.)	advection rate/ dispersion rate = $v_x L_x / D_i$	advection is slower than dispersion	dispersion is slower than advection
	St (Stanton No.)	mass-transfer rate/advection rate = $K_{l,i} L_x / v_x$ [a]	mass-transfer (dissolution) is slower than advection	advection is slower than dissolution from NAPL
Dimensionless numbers comparing biodegradation and mass transfer rates	Da_2 (Damköhler No. 2)	biodegradation rate/advection rate = $(q_{max} X_0 L_x)/(A_{eq,i} v_x)$	biodegradation is slower than advection (little potential for bioenhancement)	slow advection of substrate limits biodegradation
	Da_3 (Damköhler No. 3)	biodegradation rate/NAPL dissolution rate = $(q_{max} X_0)/(K_{l,i} A_{eq,i})$	biodegradation is slower than dissolution (little potential for bioenhancement)	slow dissolution of contaminant from NAPL limits biodegradation
	Da_6 (Damköhler No. 6)	biodegradation rate/ dispersion rate = $(q_{max} X_0 L_x^2)/(D_i A_{eq,i})$	biodegradation is slower than dispersion (little potential for bioenhancement)	slow dispersion of substrate limits biodegradation

where K is the hydraulic conductivity, and h is the hydraulic head. This linear PDE was also solved using the same FVM technique. A harmonic average was applied to calculate the hydraulic conductivity used for the Darcy flux calculation at each face of the control volumes. Following the approach of Chu et al. (2003), the pore was assumed to be occupied by either moving water, or attached biomass and associated immobile water, so:

$$n = n_w + n_b \tag{4.4}$$

where n is the porosity of the medium, n_b is the porosity occupied by biomass, and n_w is the porosity occupied by mobile water. To account for potential biomass induced

changes, the hydraulic conductivity, K, was assumed to be a function of n_b. Two different relationships between hydraulic conductivity and n_b were used, depending on whether the biomass was in the form of a smooth biofilm, or aggregate plugs in the interstitial pore space (Chu et al. 2003); however, all results presented here used the biofilm solution.

The system of biological ODEs is solved using Cash-Karp Runge-Kutta 5(4) (Chapra and Canale 2010). This is a fourth-order Runge-Kutta method with an embedded fifth-order corrector.

Several model validation steps were conducted. For example, the OS technique was tested against a first-order decay analytical solution of Bear (1972), with the FVM results using OS comparing very well with that calculated using the analytical solution (maximum relative error within the domain, $(S-S_{analytical})/S_0 < 0.0001$). Further reductions in error were obtained for smaller values for Δx and Δt. In the work reported here, the operator-splitting time step was $1/128^{th}$ of an hour. Additional model verification was performed by comparing the PCE concentration predicted by the numerical model at each location within the domain with that predicted by a steady-state two-dimensional analytical solution for pool dissolution, which neglects longitudinal dispersion (Seagren et al., 1994). The initial comparison was conducted using a uniformly spaced grid of CVs for the numerical model, which either produced a poor match to the analytical solution if the grid was too large, or was too computationally expensive if the grid was too fine. Subsequently, a varied grid was introduced, with the size of the CVs increasing moving vertically away from the DNAPL pool interface, which produced results that more closely matched the analytical solution. The most significant discrepancies were at the upstream end of the plume, where dispersion in the upstream direction was allowed in the numerical model, but not represented in the analytical solution.

Model Application: The model was used to quantify the effect of competitive interactions on DNAPL dissolution bioenhancement and plume detoxification for the experimental conditions summarized in Table 4.2 and the biokinetic parameters summarized in Table 4.3. Using these parameters, nine different scenarios were simulated transiently (Table 4.2) to evaluate the effects of biological activity on DNAPL dissolution under two sets of hydrodynamic conditions. The simulations were run with a flowrate (Q) within the micromodel of either 16 μL/hr (initial $v_x = 0.0234$ m/hr) or 160 μL/hr (initial $v_x = 0.234$ m/hr). Simulations with initial velocities of 0.0234 m/hr and 0.234 m/hr are hereafter termed the "low" v_x case, and the "high" v_x case, respectively. To assess the effects of microbial competition on the bioenhancement of DNAPL dissolution, the scenarios with biological activity differed with respect to the populations of dehalorespirers that were present. In all cases, each population was assumed to initially be present at 0.15 mg VSS/L, and the maximum biomass density allowed was set at 8000 mgVSS/L. H_2 and acetate are common intermediates of anaerobic metabolism and were provided as electron donors in varying amounts. Specifically, the competition effects were evaluated under intrinsic bioremediation conditions (low influent levels of electron donors, with $H_2 = 10$ μM, and acetate = 8 μM), and under conditions that

correspond to a biostimulation scenario (high influent electron donor concentrations, with H_2 = 600 μM, and acetate = 5000 μM). All scenarios were run for model simulation times of 672 hours (28 days). In general, quasi-steady-state conditions were achieved within this timeframe.

Table 4.2. Conditions used in Matlab modeling of experimental conditions.

Experiment	Scenario	Experimental Conditions		
		Initial Pore Water Velocity, v_x, (m/h)	**Electron donor**	**Populations Present**
1	Abiotic	1a: 0.0234; 1b: 0.234	None	None
2	*Dsm.* Pure Culture, Intrinsic	2a: 0.0234; 2b: 0.234	8 μM Acetate	*Dsm. michiganensis*
3	*Dhc.* Pure Culture, Intrinsic	3a: 0.0234; 3b: 0.234	10 μM H_2	*Dhc. mccartyi* 195
4	*Dsm.* Pure Culture, Stimulated	4a: 0.0234; 4b: 0.234	5 mM Acetate	*Dsm. michiganensis*
5	*Dhc.* Pure Culture, Stimulated	5a: 0.0234; 5b: 0.234	600 μM H_2	*Dhc. mccartyi* 195
6	Scenario 1, Intrinsic	6a: 0.0234; 6b: 0.234	8 μM Acetate 10 μM H_2	*Dsm. michiganensis*, *Dhc. mccartyi* 195
7	Scenario 1, Stimulated	7a: 0.0234; 7b: 0.234	5 mM Acetate 600 μM H_2	*Dsm. michiganensis*, *Dhc. mccartyi* 195
8	Scenario 2, Intrinsic	8a: 0.0234; 8b: 0.234	10 μM H_2	Methanogen *Dhc. mccartyi* 195
9	Scenario 2, Stimulated	9a: 0.0234; 9b: 0.234	600 μM H_2	Methanogen *Dhc. mccartyi* 195

Table 4.3. Biokinetic parameter values for microbial populations used in Matlab simulations.

Constant	*Dhc. mccartyi* 195	*Dsm. michiganensis*
q_{max} (μmol/mg VSS/h)		
PCE	6.8	12.6
TCE	7.9	17.0
DCE	13.2	
VC	0.9	
H_2	12.8	
Acetate		4.66
$K_{S,donor}$ (μM)	0.03	5.8
$K_{S,chloroethene}$ (μM)		
PCE	21.5	9.31
TCE	29.0	2.83
DCE	33.6	
VC	637	
Y (mg VSS/μmol donor)	0.0047	0.0033
k_d (h^{-1})	0.004	0.0054
H_2 threshold (μM)	0.00187	

4.2 Experimental Materials and Methods

Micromodel Design and Construction: The micromodel (Figure 4.2) includes a 5 cm x 5 cm homogeneous network of cylindrical posts 300 μm in diameter, with 173 μm pore space, 35 μm pore throat, and 35 μm depth. The pore network is connected to two inlet channels (A, B) and one outlet channel (C). One side of the pore network is connected to an open channel 5 cm in length and 1 mm in width, with an inlet (D) and an outlet (E). The micromodel was fabricated using photolithography and plasma drying etching methods, as described in previous studies (e.g., (Chomsurin and Werth 2003; Willingham et al. 2010; Willingham et al. 2008)). Briefly, the design pattern prepared in AutoCAD was first printed to a soda lime photomask using a direct-write lithography system (Intelligent Micro Patterning Inc., St Petersburg, FL). Next, a prime-grade silicon wafer 100 mm in diameter, 500 mm in thickness (Virginia Semiconductor Inc., Fredericksburg, VA) coated with a thin photoresist (PR) layer was placed under the photomask and exposed to UV light for 20 s. The area of PR exposed to UV was weakened and removed by a developer solution. Then, the exposed area of the silicon wafer was dry etched to the desired depth by the ICP-DRIE (Inductively Coupled Plasma Deep Reactive Ion Etching) method. The remaining PR was removed using a PR stripping solution, and all inlets and outlets were drilled through the wafer using a 1.25 mm diameter diamond plated drill. The finished wafer was thoroughly cleaned using acetone, isopropanol, a piranha solution (H_2SO_4: 30% H_2O_2 in 3:1 ratio) and deionized (DI) water before anodically bonding to a Pyrex glass wafer (100 mm diameter, 500 mm thickness). Finally, nanoport connectors (IDEX, Oak Harbor, WA) were attached to all inlet and outlet ports using epoxy adhesive through a thermal bonding procedure.

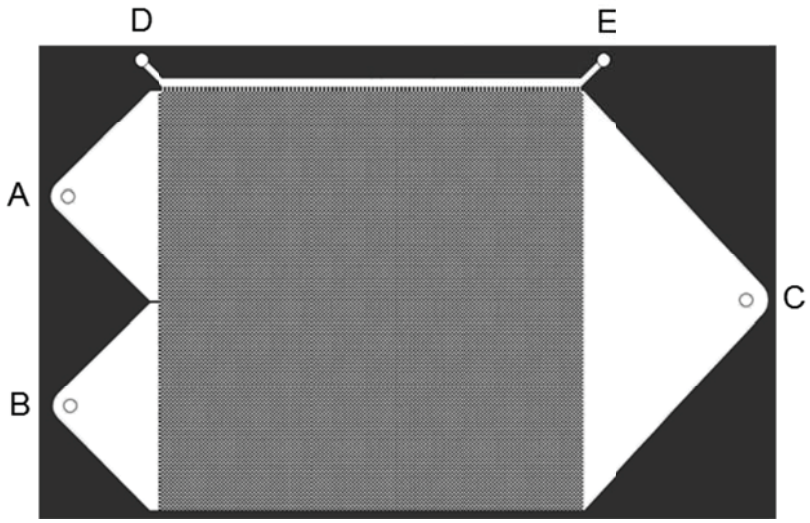

Figure 4.2. Schematic of the micromodel reactor.

Parameter Estimation: When considering DNAPL pool dissolution in the model system, the controlling parameter is the transverse hydrodynamic dispersion coefficient, D_y. Therefore, parameter estimation focused on estimating the transverse dispersivity. To prevent flow short-circuiting during these experiments, hexadecane was injected into the NAPL channel until it was full. A 20 μM fluoroscein tracer solution was then injected into the inlet away from the NAPL channel (inlet B in Figure 4.2), and nano-pure water (NPW) was injected at the same flow rate in the other inlet (inlet A in Figure 4.2). The tracer and water were allowed to flow until a steady-state was reached, which was assumed to occur after more than 5 pore volumes of flow had passed through the micromodel, and confirmed visually. This procedure was repeated for four flow rates. At each flow condition, a series of at least three large images were collected using fluorescent microscopy, and image analysis was used to fit a value of transverse dispersion at the center of each domain following the approach of Willingham et al. (2010; 2008). Briefly, using fluorescence data from between the grains, an analytical solution to the AD equation was fit to the data using the built-in least-squares curve fitting routine (lsqcurvefit) in Matlab, with default settings. The best-fit transverse hydrodynamic dispersion coefficient was modeled as the sum of effective molecular diffusion and mechanical dispersion, given in simplified form as:

$$D_y = \alpha_t v_x + \tau d_i \tag{4.5}$$

where α_t is the transverse dispersivity; τ is the tortuosity; and d_i is the aqueous molecular diffusion coefficient. The value of τ was estimated to be 0.1297 based on the porosity (Domenico and Schwartz 1998), and the aqueous molecular diffusion coefficient (d_i) was calculated using the Hayduk-Laudie equation.

Longitudinal Dispersivity and Porosity: The longitudinal dispersivity, α_L, and porosity of the micromodel porous medium were estimated following the approach of Willingham et al. (2010). Specifically, α_L was set equal to the porous medium grain diameter = 300 μm. This assumption is consistent with the work of Rumer (1962) who reported that the value of α_L should be of the same order of magnitude as the mean grain size. The porosity was calculated directly from the geometry of the micromodel porous medium design, n = 0.39.

4.3 Results and Discussion

Parameter Estimation and Abiotic Dissolution Rate: Transverse hydrodynamic dispersion coefficient (D_y) values were estimated using a non-reactive tracer and fluorescent microscopy, as described above. At each flow rate, a D_y value was fit to fluorescence data measured across a cross section of the flow cell. An example of the fluorescence data obtained during a tracer study is shown in Figure 4.3, along with the best fit of the analytical solution to these data. The D_y values obtained at each flow rate are summarized in Table 4.4. Using Eq. (4.5) and the parameter values reported above, a least-squares regression was used to fit a value of α_t = 8.09 μm to the dispersion coefficient data as a function of v_t. The magnitude of α_t indicates that transverse mechanical dispersion is a weak process in the flow-cell experimental system compared to longitudinal dispersion (α_L = 300 μm). Based on the estimated dispersivity values and the dimensionless number definitions in Table 4.1, the longitudinal Peclet number, Pe_l, is initially 1.6×10^2 and 1.7×10^2, under the low and high velocity conditions, respectively, and the transverse Peclet number, Pe_t is initially 1.9×10^3 and 5.1×10^3, under the low and high velocity, respectively. Following the approach of Johnson et al. (2013), with the assumption of local equilibrium at the DNAPL/aqueous interface, these values indicate that hydrodynamic dispersion, in particular, transverse dispersion is the rate-limiting mass-transfer process in the system.

As discussed by Seagren et al. (1994, 1999), for an idealized NAPL pool, the dispersive mass flux, J [$M_S L^{-2} T^{-1}$], across a unit area of the NAPL/aqueous phase interface (y=0) can be described using a relationship analogous to Fick's first law, as long as the advective mass flux is only parallel to the NAPL interface (Freeze and Cherry 1979):

$$J = -nD_y \left(\frac{\partial A_i}{\partial y} \right)_{y=0} \tag{4.6}$$

When the interphase mass-transfer rate described by Eq. (4.6) is much faster than advective-dispersive and reactive solute sinks in the domain near the interface, local thermodynamic equilibrium can exist at the interface, as is assumed in this work.

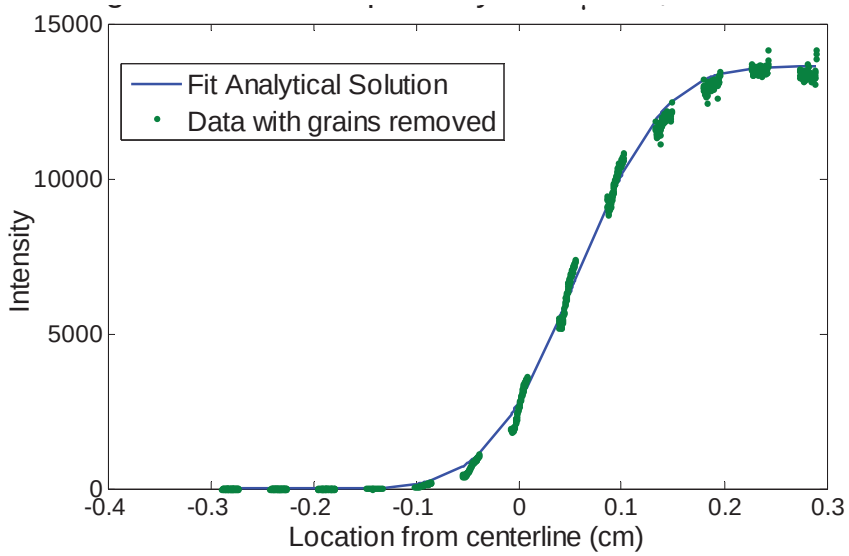

Figure 4.3. Representative fluorescent tracer study data used to fit D_y.

Table 4.4. Transverse hydrodynamic dispersion coefficient (D_y) values estimated from tracer studies conducted at different velocity (v_x) values.

Flow rate (µL/hr)	**v_x (cm/hr)**	**D_y (cm^2/hr)**			
		Replicate 1	Replicate 2	Replicate 3	Average
1000	146.5	0.125	0.135	0.138	0.133
640	93.8	0.0816	0.0788	0.0700	0.0768
160	23.4	0.0217	0.0207	0.0176	0.0200
80	11.7	0.0103	0.0110	0.0109	0.0107

Assuming equilibrium exists at the interface, the dissolution mass flux away from the interface can be enhanced by increasing D_y and /or the concentration gradient in Eq. (4.6). Increasing the groundwater flow rate can increase D_y because it is a function of v_x (Eq. 4.5), and the concentration gradient because of the increased advective solute sink term, i.e., A_i is lowered by the flushing out of dissolved solute. For example, under the conditions used in these simulations, the model predicts an abiotic PCE dissolution rate of 3.7×10^{-4} µmol/hr under the low velocity conditions (Experiment 1a), which increases to 2.3×10^{-3} µmol/hr at the high velocity (Experiment 1b). These values were confirmed to be consistent with the abiotic dissolution model of Seagren et al. (1994).

Single bacterial population simulations. Biodegradation can also increase the concentration gradient in Eq. (4.6) by being a reaction sink that decreases A_i. The bioenhancement of the dissolution flux in Eq. 4.6 can be quantified as the bioenhancement factor, E, which is defined here as the ratio of the DNAPL source dissolution mass rate with biodegradation, R_{biotic}, to the dissolution rate without biodegradation, $R_{abiotic}$ (Seagren et al., 1994; Becker and Seagren, 2009):

$$E = \frac{R_{biotic}}{R_{abiotic}} \tag{4.7}$$

To examine this bioenhancement effect, as well as any resulting plume detoxification, *Dhc. mccartyi* 195 and *Dsm. michiganensis* were each first simulated in pure culture before being simulated in competition. Several key observations related to dissolution bioenhancement and PCE detoxification can be made by comparing the results of these pure culture simulations with the abiotic dissolution simulation results. Under intrinsic (low) electron donor levels, with *Dsm. michiganensis* alone (Experiments 2a and 2b), or *Dhc. mccartyi* 195 alone (Experiments 3a and 3b), there was little biotransformation of the PCE, and accordingly, the biotic dissolution flux was similar to that under abiotic conditions at the low and high velocities, with $E \approx 1.0$ (data not shown). This is consistent with the magnitude of the biodegradation rate compared to the advection rate, as reflected by Da_2, and the magnitude of the biodegradation rate compared to the dispersion rate, as reflected by Damköhler No. 6, Da_6 (Table 4.1). Specifically, Da_2 was on the order of 10^{-3} and 10^{-4} under the low and high velocity conditions, respectively, for *Dhc. mccartyi* 195 and *Dsm. michiganensis*, assuming an $A_{eq,i}$ equal to the PCE solubility and the initial biomass conditions. Similarly, using the same assumptions, $10^0 \leq Da_6 < 10^1$ for *Dhc. mccartyi* 195 and *Dsm. michiganensis*, with dispersion the overall rate limiting process under the low velocity conditions, but both biodegradation and dispersion rate limiting under the high velocity conditions. Accordingly, biodegradation was the overall rate-limiting process for both populations, or at least similar to or as rate limiting as mass transfer via dispersion, and biodegradation was expected to have little or no effect on the dissolution rate. Consistent with this conclusion, although biomass levels did increase over time for *Dhc. mccartyi* 195 and *Dsm. michiganensis* under the low and high velocity conditions, the impact on the mass rate of electron donor and acceptor exiting the domain was negligible. Becker and Seagren (2009) saw similar results in their simulations with pure cultures of *Dhc. mccartyi* 195 and *Dsm. michiganensis* in a one-dimensional domain with PCE blobs, under intrinsic electron donor levels, and low and high velocities.

The mass rate of ethenes exiting the domain for Experiment 5a with *Dhc. mccartyi* 195 under engineered (stimulated) conditions at the low v_x are presented in Figure 4.4. Initially, as in the intrinsic case, there is no bioenhancement. However, in this case, there is a relatively rapid increase in total ethenes leaving the system between 100 and 150 hrs, as the biomass increased (Figure 4.5a) as a result of the higher electron donor

concentrations. The total ethenes efflux rate subsequently plateaus. Although the total ethenes efflux rate is fairly constant, there is a steadily increasing rate of ethene leaving the system. The latter is important because ethene is a non-toxic product of PCE dechlorination and a desirable bioremediation end-product. In addition, at the end of the 672 hr (28 d) simulation, the approximately 4.50 $x10^{-4}$ μmol/hr total ethenes mass rate represents a small enhancement ($E \approx 1.2X$) of the total dissolution rate observed under abiotic conditions. Consistent with these results, analysis of the dimensionless parameters indicates that conditions were suitable for bioenhancement because biodegradation was limited by advection (i.e., electron donor supply), as indicated by $Da_2 >> 1$, and transverse dispersion (i.e., electron acceptor supply), as per $Da_6 >> 1$,based on the highest biomass levels achieved in the domain and assuming an $A_{eq,i}$ equal to the PCE solubility.

The total ethene mass rate and distribution at the end of the model simulation for *Dhc. mccartyi* 195 under engineered bioremediation and low v_x conditions (Experiment 5a) are compared in Figure 4.6 to the low v_x results for *Dhc. mccartyi* 195 under intrinsic conditions (Experiment 3a), and abiotic conditions (Experiment 1a). Clearly, biostimulation resulting from high concentrations of electron donor (600 μM H_2) increased dechlorination of PCE by *Dhc. mccartyi* 195 and enhanced dissolution compared to under intrinsic conditions as indicated by an increase in the total ethene mass rate. In addition, the excess electron donor resulted in dechlorination of PCE to lesser chlorinated ethenes (including vinyl chloride and ethene).

The simulation results for the mass rate of ethenes exiting the domain for Experiment 4a with *Dsm. michiganensis* under engineered, low v_x conditions are presented in Figure 4.7. Two key observations can be made by comparing Figures 4.4 and 4.7. First, in the presence of *Dsm. michiganensis* alone, dichloroethene was the dominant dechlorination product. Second, bioenhancement of dissolution under biostimulation conditions was nine times higher in the presence of *Dsm. michiganensis* compared with *Dhc. mccartyi* 195 ($E \approx 9.9$), based on the total ethenes efflux rate. Two factors most likely contributed to the greater bioenhancement in the presence of *Dsm. michiganensis*. First, *Dsm. michiganensis* has faster PCE utilization kinetics compared with *Dhc. mccartyi* 195 (Table 4.3). Rapid depletion of PCE in the aqueous phase creates a strong driving force for dissolution from the DNAPL phase (Eq. (4.6)). Second, unlike the *Dhc. mccartyi* 195 biomass, which arched away from the PCE plume with increasing distance from the inlet, *Dsm. michiganensis* growth was concentrated along the DNAPL water interface where it could directly impact PCE dissolution (Fig. 4.5b). A previous study showed that dechlorinating populations grow toward the limiting substrate (Chu et al. 2003). Thus, the rapid depletion of aqueous-phase PCE and growth of *Dsm. michiganensis* along the interface clearly show that *Dsm. michiganensis* was limited by the availability of PCE in the current study. The pattern of *Dhc. mccartyi* 195 growth away from the PCE- or DNAPL-water interface suggests that it was limited by H_2.

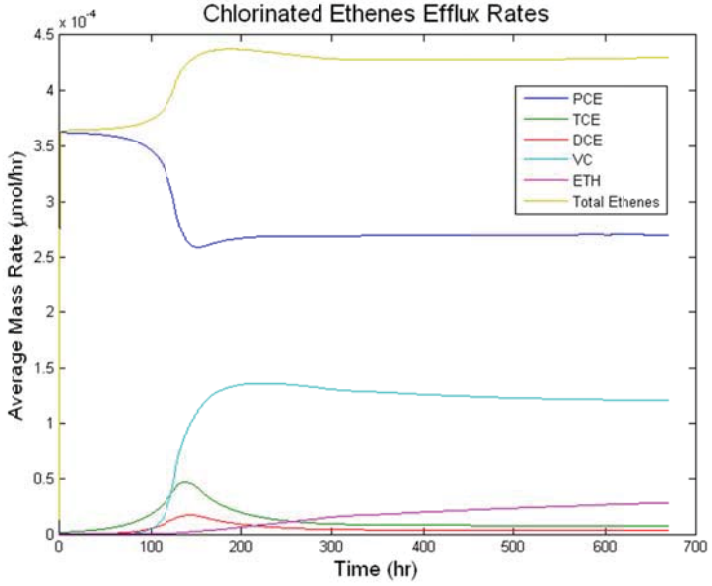

Figure 4.4. Mass rate of ethenes exiting the domain for Experiment 5a with *Dhc. mccartyi* 195 under engineered conditions and low v_x.

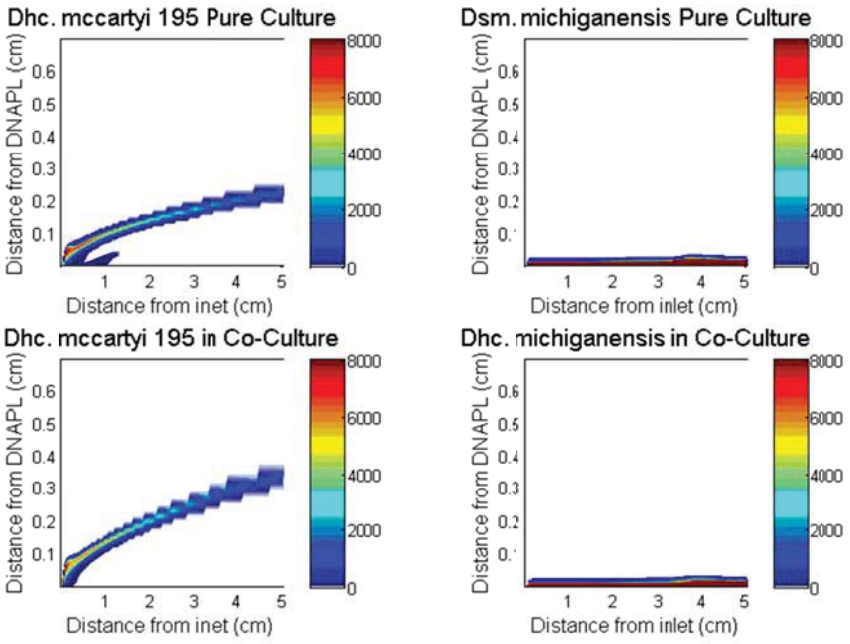

Figure 4.5. Biomass concentration at 14 d under low velocity, biostimulated conditions for pure cultures of *Dhc. mccartyi* 195 and *Dsm. michiganensis* in pure culture (top row) and individual populations in a co-culture of *Dhc. mccartyi* 195 and *Dsm. michiganensis* (bottom row). The portion of the domain within 0.7 cm of the DNAPL is shown because only this region contained significant biomass. The scale on the right is given in terms of mg VSS/L.

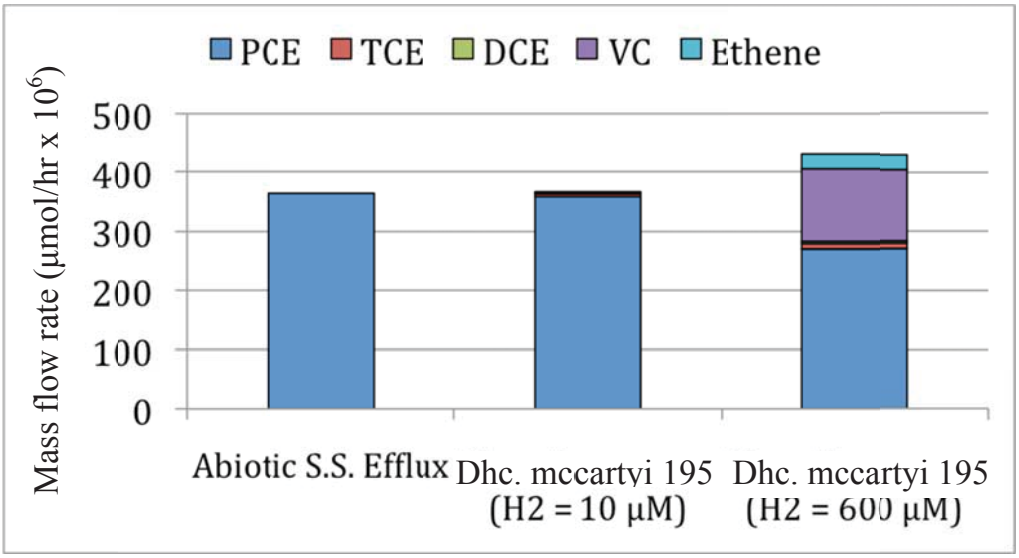

Figure 4.6. Mass rate of chlorinated ethenes and ethene leaving simulation domain under abiotic, low v_x conditions and with *Dhc. mccartyi* 195 present under intrinsic bioremediation (10 µM H_2) and engineered bioremediation (600 µM H_2) conditions.

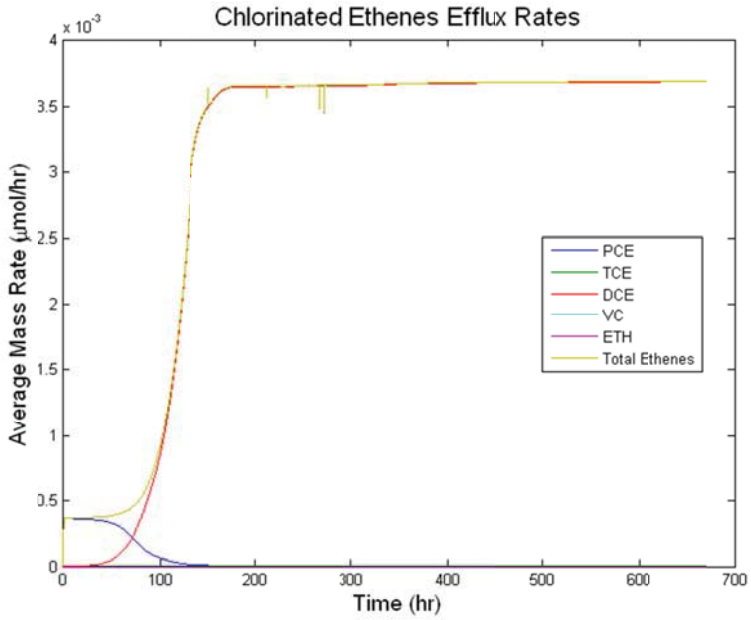

Figure 4.7. Mass flow rate of ethenes exiting the domain for Experiment 4a with *Dsm. michiganensis* under engineered conditions and low flow.

Similar results were observed under engineered conditions at the high v_x, as under engineered low velocity conditions, but the total dissolution rate increased due to the faster velocity (Eq. 4.6), while the bioenhancement effect is reduced because the biodegradation rate is becoming smaller relative to flow rate. The biomass distribution was also affected as the velocity was increased, due to the dissolved contaminant plume being tighter to pool. Accordingly, the *Dhc. mccartyi* 195 biomass migrates toward the pool (Figure 4.8), while the increased dissolution rate resulted in more growth of *Dsm. michiganensis*.

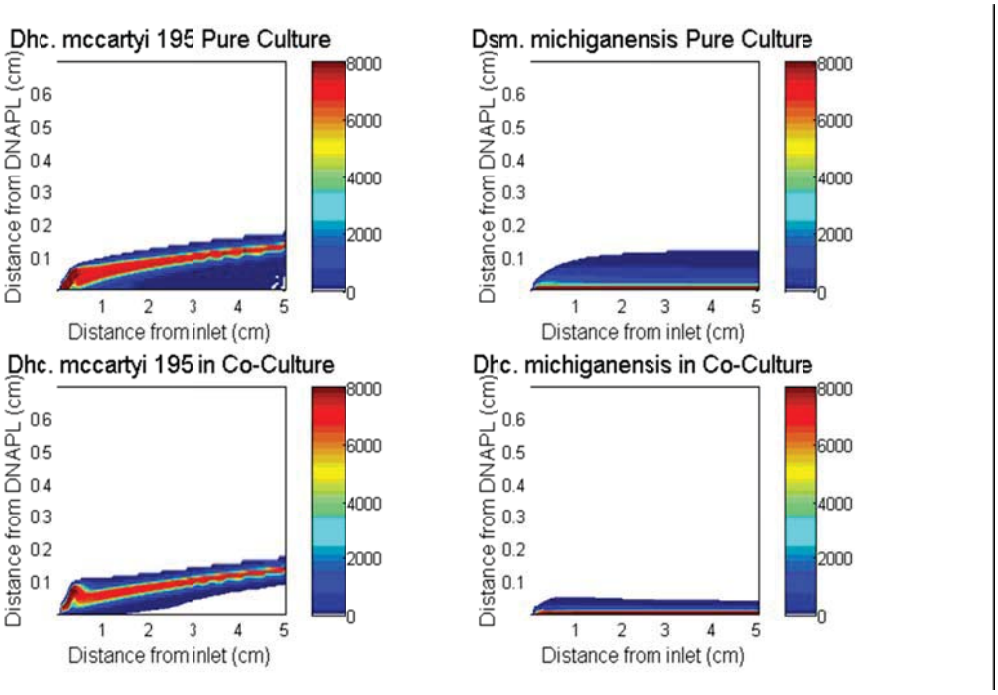

Figure 4.8. Biomass concentration at 14 d under high velocity, biostimulated conditions for pure cultures of *Dhc. mccartyi* 195 and *Dsm. michiganensis* in pure culture (top row) and individual populations in a co-culture of *Dhc. mccartyi* 195 and *Dsm. michiganensis* (bottom row). The portion of the domain within 0.7 cm of the DNAPL is shown because only this region contained significant biomass. The scale on the right is given in terms of mg VSS/L.

Competition simulations. Similar to the simulations with the individual populations, under intrinsic (low) electron donor levels, with *Dsm. michiganensis* in competition with *Dhc. mccartyi* 195 (Experiments 6a and 6b), there was little biotransformation of the PCE, and accordingly, the biotic dissolution flux was similar to that under abiotic conditions at the low and high velocities (data not shown). As discussed above, this is consistent with the magnitude of the biodegradation rate compared to the advection rate, as reflected by Da_2, and the magnitude of the biodegradation rate compared to the dispersion rate, as reflected by Da_6. Accordingly, biodegradation was the overall rate-

limiting process for both populations, or at least similar to or as rate limiting as mass transfer, and biodegradation was expected to have little or no effect on the dissolution rate with biomass levels similar to the initial conditions, as observed.

As for the individual populations, more dramatic effects are observed under the engineered conditions, because of the faster biomass growth and biotransformation rates resulting from the higher electron donor concentratons. The mass rates of the ethenes exiting the domain in Scenario 7a, in which *Dhc. mccartyi* 195 and *Dsm. michiganensis* compete under biostimulation and low v_x conditions are shown in Figure 4.9. Several important findings emerge by comparing the results in Figure 4.9 to those obtained with *Dsm. michiganensis* (Figure 4.7) or *Dhc. mccartyi* 195 (Figure 4.4) alone. One key finding is the effect of competition on the total mass rate of ethenes leaving the domain. The total ethene mass rate exiting the domain populated by *Dsm. michiganensis* alone is approximately 3.7 x 10^{-3} µmol/hr (Figure 4.7). However, the total ethene mass rate increased to 4.0 x 10^{-3} µmol/hr (E ≈ 10.8) when the domain was populated by *Dsm. michiganensis* in co-culture with *Dhc. mccartyi* 195, which alone achieved a total ethene

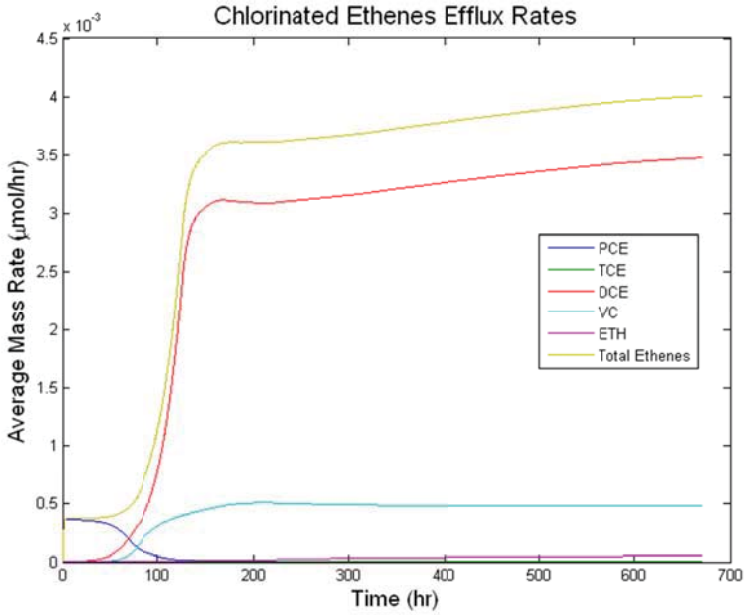

Figure 4.9. Mass flow rate of ethenes exiting the domain for Experiment 7a with *Dhc. mccartyi* 195 and *Dsm. michiganensis* under engineered conditions and low flow.

mass rate of approximately 4.3 x 10^{-4} µmol/hr. These results suggest that the roughly ten-fold increase in dissolution resulting from the rapid utilization of aqueous-phase PCE by *Dsm. michiganensis* was modestly enhanced by the activity of *Dhc. mccartyi* 195, and highlight the role that competition can play in determining the amount of bioenhancement that can occur. As noted above for the single population simulations, analysis of the dimensionless parameters indicates that the low v_x, engineered conditions were suitable for bioenhancement because biodegradation was limited by advection (i.e., electron

donor supply), as indicated by Da_2>>1, and transverse dispersion (i.e., electron acceptor supply), as per Da_6>>1,based on the highest biomass levels achieved in the domain and assuming an $A_{eq,i}$ equal to the PCE solubility.

Although the increase in the dissolution bioenhancement with the co-culture under biostimulation and low v_x conditions was relatively modest, the distribution and relative mass rates of the chlorinated ethenes exiting the domain shows that having multiple dehalorespirers that utilize different electron donors can greatly enhance the extent of PCE transformation that is achieved in situ, as previously suggested by Becker (2006) and Becker and Seagren (2009). For example, with *Dhc. mccartyi* 195 alone (Figure 4.4), the dominant chlorinated ethenes exiting the domain at steady state are PCE followed by VC, whereas with *Dsm. michiganensis* alone (Figure 4.7), the dominant chlorinated ethene is DCE (essentially 100%). In comparison, in co-culture (Figure 4.9) the dominant intermediates exiting the domain at the end of the model run were DCE followed by VC, indicating a greater extent of PCE transformation compared to either population alone. These observations can be explained as follows. Growing alone, *Dhc. mccartyi* had to invest 3 mol H_2/mol PCE to synthesize VC. Because it was limited by H_2 availability, as discussed above, this meant that the aqueous PCE concentration remained relatively high and limited VC and ethene production. In contrast, when grown in the co-culture, *Dsm. michiganensis* reductively dechlorinated PCE to DCE using electron equivalents derived from acetate. Therefore, *Dhc. mccartyi* 195 invested electron equivalents derived from H_2 in dechlorination of the lesser chlorinated ethenes and greater net transformation of PCE to VC was observed.

As seen with the pure culture simulations under biostimulation and low v_x conditions, the two species in co-culture both grow near the DNAPL, but the proximity of the highest concentration of biomass to the DNAPL pool is key to the enhancement observed (Figure 4.5). Compared to the pure culture simulation (Figure 4.5b), the location of the *Dsm. michiganensis* biomass was not significantly influenced by the presence of *Dhc. mccartyi* 195 (Figure 5d), but the converse was not true. Compared to its pure culture behavior (Figure 4.5a), *Dhc. mccartyi* 195 grew further away from the DNAPL source when grown in co-culture (Figure 4.5c), suggesting that in this scenario, it was relying on the DCE produced by *Dsm. michiganensis*, not the PCE from the DNAPL source, for growth. Becker and Seagren (2009) observed some similar spatial differences in their three-cells-in-series with NAPL blobs model used in that work. In that model, *Dhc. mccartyi* 195 eventually outcompeted *Dsm. michiganensis* for chlorinated ethenes in the first cell, with its activity controlled by the H_2 input into the domain, whereas *Dsm. michiganensis* dominated in the second cell, growing on the acetate. The key difference with the current work is that in Becker and Seagren's (2009) model, the DNAPL blobs were originally distributed throughout the domain, allowing growth of both populations throughout the domain. In comparison, in the current work, the DNAPL source is only present on one side of the domain. Although both populations were initially uniformly distributed, the ability of *Dsm. michiganensis* to use acetate as an electron donor in regions where H_2 levels were depleted, coupled with its faster kinetics, resulted in it being the population that grew nearest the DNAPL source and primarily responsible for bioenhancement of

DNAPL dissolution. The improved spatial modeling of the competing populations may also explain why the levels of bioenhancement predicted in this work are closer than those of Becker and Seagren (2009) to the levels previously observed in laboratory column and tank studies at relatively slow average pore water velocities with mixed microbial cultures (E ≈ 2 to 13) (Da Silva et al. 2006; Glover et al. 2007; Sleep et al. 2006; Yang and McCarty 2000; Yang and McCarty 2002).

The co-culture simulations also highlight the impact of hydrodynamic conditions on the outcome of the population interactions. Most importantly, v_x has a major impact on the relative abundance of each population , as illustrated in Figures 4.10 and 4.11, and in turn dissolution bioenhancement and plume detoxification. Under the low v_x (Experiment 7a) conditions, *Dsm. michiganensis* is initially dominant given its kinetic competitive advantage as reflected in the q_{max}/K ratio (Healey 1980), with $q_{max}/K_{PCE} = 1.4$ and $q_{max}/K_{TCE} = 6.0$, compared to $q_{max}/K_{PCE} = 0.32$ and $q_{max}/K_{TCE} = 0.27$ for *Dhc. mccartyi* 195 (Table 4.3). As a result of this early growth, the bioenhancement factor, E is approximately 9.7 by 200 hours (similar to that achieved with *Dsm. michiganensis* alone), and *Dsm. michiganensis* has nearly twice as much total biomass at 672 hours (28 days) compared with *Dhc. mccartyi* 195 (Figure 4.10). Nevertheless, although the *Dhc. mccartyi* 195 biomass is relatively small, it still has an impact on detoxification and bioenhancement. As the *Dhc. mccartyi* 195 biomass starts to grow initially, the impact on detoxification is clear, as discussed above: whereas with *Dsm. michiganensis* alone the only chlorinated ethane exiting the domain was DCE, in co-culture (Figure 4.9) the dominant intermediates exiting the domain at the end of the model run were DCE (roughly 69% of the total mass flow of ethenes) followed by VC (roughly 30% of the mass flow). Subsequently, the *Dhc. mccartyi* 195 biomass continues to increase due to its relatively high yield coefficient (Table 4.3), and the steady production of DCE by *Dsm. michiganensis*. As a result there is an increase in the bioenhancement factor, E, to 10.8, with slowly declining levels of VC, and slowly increasing levels of ethane, which reached 1.3% of the total ethenes mass flow rate by the end of the simulation.

In comparison, under the high v_x conditions (Experiment 7b), the general patterns are similar to the low v_x case (Figure 4.12), but there is a switch in the dominant population. While *Dsm. michiganensis* is again initially dominant, *Dhc. mccartyi* 195 becomes the dominant population over time, with the *Dhc. mccartyi* 195 biomass approaching almost twice the *Dsm. michiganensis* biomass by the end of the simulation (Figure 4.11). Initially *Dhc. mccartyi* 195 appears to be growing on the DCE produced by *Dsm. michiganensis*, but as the biomass increases, *Dhc. mccartyi* 195 is growing on PCE and TCE as well, at the expense of *Dsm. michiganensis*. The latter is facilitated by the migration of the *Dhc. mccartyi* 195 biomass toward the pool with the increased velocity and reduced extent of the plume into the domain (Figure 4.8). The shift in dominance to *Dhc. mccartyi* 195 results not only in an increase in the bioenhancement effect, but a shift in the chlorinated ethenes exiting the domain compared to the low v_x case, with an increase in the fraction of the total ethenes mass flow in the form of VC and ethene.

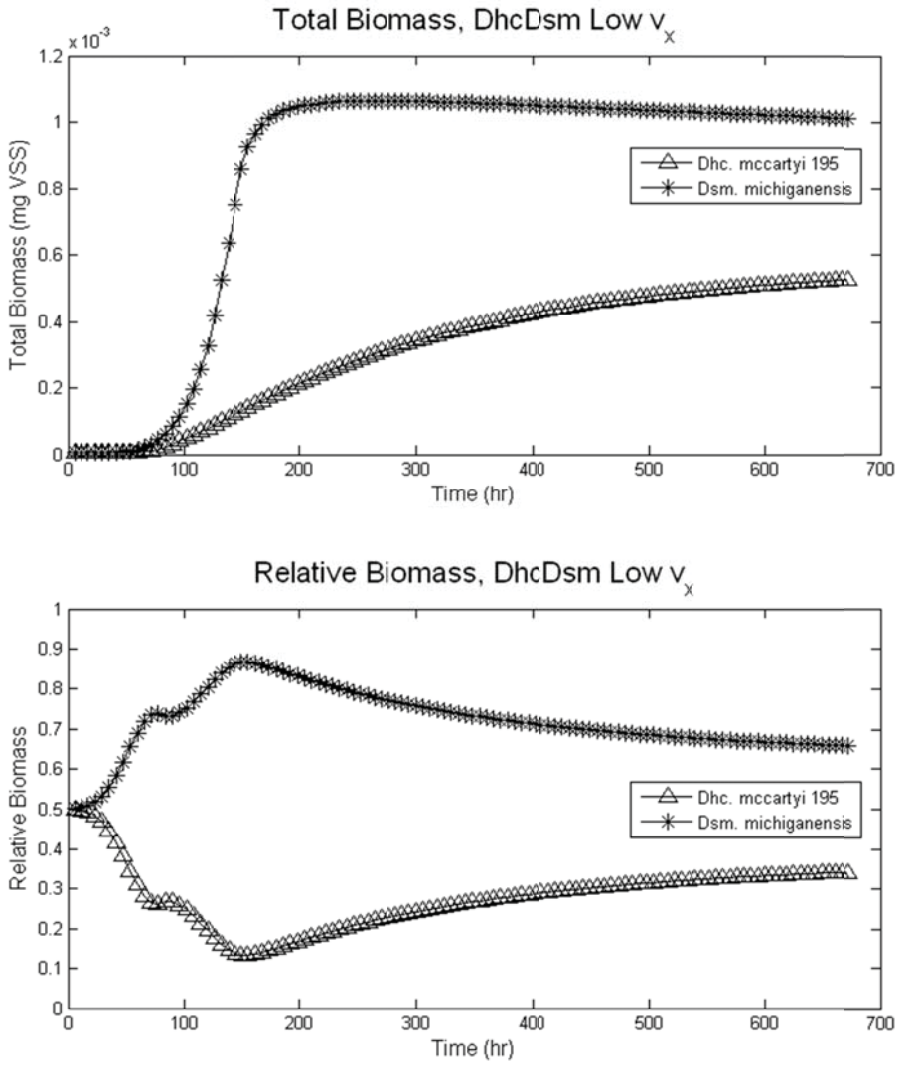

Figure 4.10. Total biomass in model domain under low velocity, engineered conditions: (a) total mass (mg VSS) of *Dhc. mccartyi* 195, and *Dsm. michiganensis* in domain, (b) fraction of mass of *Dhc. mccartyi* 195 and *Dsm. michiganensis* relative to the total biomass.

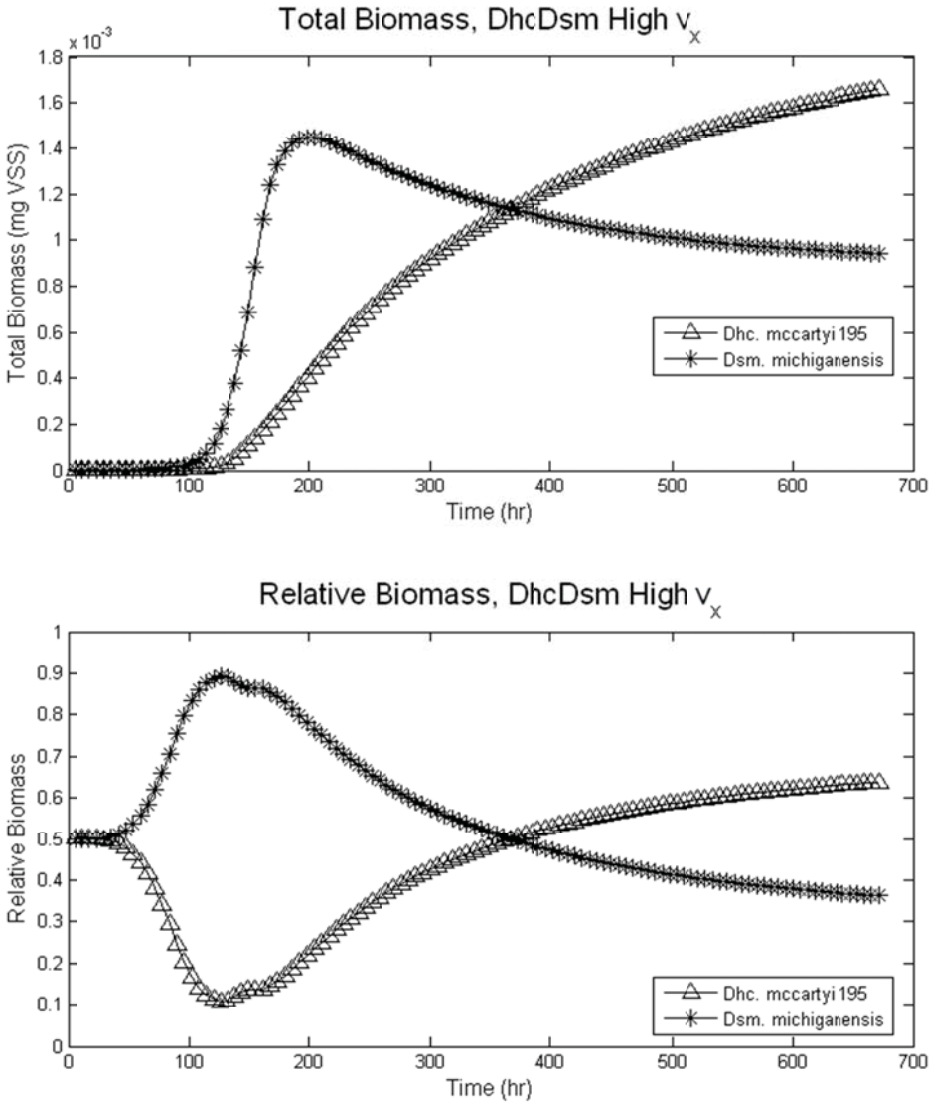

Figure 4.11. Total biomass in model domain under high velocity, engineered conditions: (a) total mass (mg VSS) of *Dhc. mccartyi* 195, and *Dsm. michiganensis* in domain, (b) fraction of mass of *Dhc. mccartyi* 195 and *Dsm. michiganensis* relative to the total biomass.

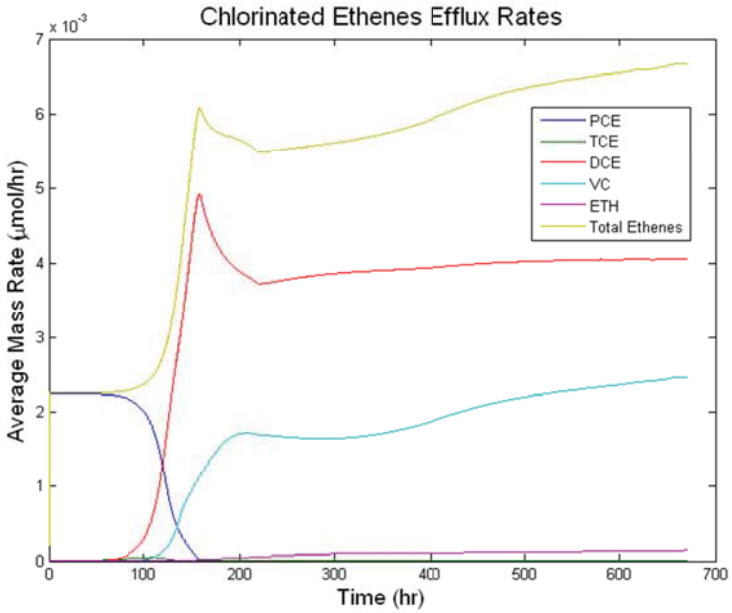

Figure 4.12. Mass flow rate of ethenes exiting the domain for Experiment 7b with *Dhc. mccartyi* 195 and *Dsm. michiganensis* under engineered conditions and high flow.

4.4 Conclusions

These modeling results illustrate some of the complexities involved in understanding the outcome of microbial competition for electron acceptor substrates at chlorinated ethene DNAPL sites. Clearly the outcome of competition and the resulting effects on plume detoxification and dissolution bioenhancement are a function of not only the population's affinity for the substrates, as represented by the ratio q_{max}/K, but also the relative magnitude of the yield coefficients, the rate of supply of electron donor, and the hydrodynamics. Consistent with the conclusions of Becker and Seagren (2009), these results suggest that biostimulation and bioaugmentation strategies for DNAPL source zones should focus on increasing the abundance of not only *Dehalococcoides* species, but also heterotrophic PCE-respiring populations like *Dsm. michiganensis*. As pointed out by Becker (2006), organisms like *Dsm. michiganensis* that specialize in the dechlorination of PCE to DCE (or TCE), are beneficial because of their faster kinetics compared to *Dehalococcoides* strains. In addition, this work shows that the ability of heterotrophic PCE-respiring strains like *Dsm. michiganensis* to use acetate as an electron donor in regions where H_2 levels are depleted, coupled with their faster kinetics, may result in their being the primary populations responsible for bioenhancement of DNAPL dissolution. Nevertheless, the presence of *Dhc. mccartyi* 195 is critical for increased detoxification of the dissolved plume.

5 References

AFCEE. 2004. Principles and Practices of Enhanced Bioremediation of Chlorinated Solvents (Final Report). Air Force Center for Environmental Excellence.

Aulenta F, Canosa A, Leccese M, Papini MP, Majone M, Viotti P. 2007. Field study of in situ anaerobic bioremediation of a chlorinated solvent source zone. Ind. Eng. Chem. Res. 46:6812-6819.

Bear J. 1972. Dynamics of fluids in porous media: American Elsevier Publication Company.

Becker JG. 2006. A modeling study and implications of competition between *Dehalococcoides ethenogenes* and other tetrachloroethene-respiring bacteria. Environ. Sci. Technol. 40:4473-4480.

Becker JG, Seagren EA. 2009. Modeling the Effects of Microbial Competition and Hydrodynamics on the Dissolution and Detoxification of Dense Nonaqueous Phase Liquid Contaminants. Environ. Sci. Technol. 43(3):870-877.

Carr CS, Garg S, Hughes JB. 2000. Effect of dechlorinating bacteria on the longevity and composition of PCE-containing nonaqueous phase liquids under equilibrium dissolution conditions. Environ. Sci. Technol. 34(6):1088-1094.

Carrayrou J, Mosé R, Behra P. 2004. Operator-splitting procedures for reactive transport and comparison of mass balance errors. Journal of Contaminant Hydrology 68(3-4):239-268.

Chapra SC, Canale RP. 2010. Numerical methods for engineers. Boston: McGraw-Hill Higher Education. xviii, 968 p. p.

Chomsurin C, Werth CJ. 2003. Analysis of pore-scale nonaqueous phase liquid dissolution in etched silicon pore networks. Water Resources Research 39(9).

Christ JA, Ramsburg CA, Abriola LM, Pennell KD, Löffler FE. 2005. Coupling aggressive mass removal with microbial reductive dechlorination for remediation of DNAPL source zones: A review and assessment. Environ. Health Perspect 113(4):465-477.

Chu M, Kitanidis PK, McCarty PL. 2003. Effects of biomass accumulation on microbially enhanced dissolution of a PCE pool: a numerical simulation. Journal of Contaminant Hydrology 65(1-2):79-100.

Cope N, Hughes JB. 2001. Biologically-enhanced removal of PCE from NAPL source zones. Environ. Sci. Technol. 35(10):2014-2021.

Da Silva M, Daprato R, Gomez D, Hughes J, Ward C, Alvarez P. 2006. Comparison of bioaugmentation and biostimulation for the enhancement of dense nonaqueous phase liquid source zone bioremediation. Water Environment Research 78(13):2456-2465.

Domenico PA, Schwartz FW. 1998. Physical and chemical hydrogeology. New York: Wiley. xiii, 506 p. p.

EPA. 1993. Evaluation of the Likelihood of DNAPL Presence at NPL Sites, National Results. Office of Solid Waste and Emergency Response (OSWER). Report nr EPA/540-R-93-073.

Essaid HI, Cozzarelli IM, Eganhouse RP, Herkelrath WN, Bekins BA, Delin GN. 2003. Inverse modeling of BTEX dissolution and biodegradation at the Bemidji, MN crude-oil spill site. J. Contam. Hydrol. 67:269-299.

Freeze RA, Cherry JA. 1979. Groundwater. Englewood Cliffs, NJ: Prentice-Hall.

Glover K, Munakata-Marr J, Illangasekare T. 2007. Biologically enhanced mass transfer of tetrachloroethene from DNAPL in source zones: Experimental evaluation and influence of pool morphology. Environmental Science & Technology 41(4):1384-1389.

Healey FP. 1980. Slope of the Monod equation as an indicator of advantage in nutrient competition. Microbial Ecology 5(4):281-286

Johnson MA, Song X, Seagren EA. 2013. A quantitative framework for understanding complex interactions between competing interfacial processes and in situ biodegradation. Journal of Contaminant Hydrology 146:16-36.

Knutson C, Valocchi A, Werth C. 2007. Comparison of continuum and pore-scale models of nutrient biodegradation under transverse mixing conditions. Advances in Water Resources 30(6-7):1421-1431.

Lendvay JM, Loffler FE, Dollhopf M, Aiello MR, Daniels G, Fathepure BZ, Gebhard M, Heine R, Helton R, Shi J and others. 2003. Bioreactive barriers: A comparison of bioaugmentation and biostimulation for chlorinated solvent remediation. Environmental Science & Technology 37(7):1422-1431.

Nester EW. 2009. Microbiology : a human perspective. Dubuque, IA: McGraw-Hill. 1 v. (various pagings) p.

Rumer RR. 1962. Longitudinal dispersion in steady and unsteady flow. J. Hydraul. Div. (ASCE) 88 (HY4):147-172.

Seagren EA, Moore TO. 2003. Nonaqueous phase liquid pool dissolution as a function of average pore water velocity. Journal of Environmental Engineering-Asce 129(9):786-799.

Seagren EA, Rittmann BE, Valocchi AJ. 1993. Quantitative evaluation of flushing and biodegradation for enhancing in situ dissolution of nonaqueous-phase liquids. J. Contam. Hydrol. 12:103-132.

Seagren EA, Rittmann BE, Valocchi AJ. 1994. Quantitative evaluation of the enhancement of NAPL-pool dissolution by flushing and biodegradation. Environ. Sci. Technol. 28(5):833-839.

Seagren EA, Rittmann BE, Valocchi AJ. 1999. A critical evaluation of the local equilibrium assumption in modeling NAPL-pool dissolution. Journal of Contaminant Hydrology 39(1-2):109-135.

Seagren EA, Rittmann BE, Valocchi AJ. 2002. Bioenhancement of NAPL pool dissolution: Experimental evaluation. Journal of Contaminant Hydrology 55(1-2):57-85.

Sleep B, Seepersad D, Mo K, Heidorn C, Hrapovic L, Morrill P, McMaster M, Hood E, Lebron C, Lollar B and others. 2006. Biological enhancement of tetrachloroethene dissolution and associated microbial community changes. Environmental Science & Technology 40(11):3623-3633.

Sorenson KS, Jr. 2003. Enhanced bioremediation for treatment of chlorinated solvent residual source areas. In: Henry SM, Warner SD, editors. Chlorinated Solvent and

DNAPL Remediation, Innovative Strategies for Subsurface Cleanup. Washington, DC: American Chemical Society. p 119-131.

Table 1 References:. Ave. reported values from: [1]EPA530-R-09-002, www.epa.gov/epawaste; [2]http://www.industrialresourcescouncil.org/; [3]http://minerals.usgs.gov/minerals/pubs/commodity/iron_&_steel_slag/myb1-2006-fesla.pdf; [4]http://www.epa.gov/osw/nonhaz/pdfs/biowood.pdf; [5]www.epa.gov/waste/nonhaz/pdfs/biofood.pdf.

Valocchi AJ, Malmstead M. 1992. Accuracy of operator splitting for advection-dispersion-reaction problems. Water Resources Research 28(5):1471-1476.

Versteeg HK, Malalasekera W. 2007. An introduction to computational fluid dynamics : the finite volume method. Harlow, England ; New York: Pearson Education Ltd. xii, 503 p. p.

Willingham T, Zhang CY, Werth CJ, Valocchi AJ, Oostrom M, Wietsma TW. 2010. Using dispersivity values to quantify the effects of pore-scale flow focusing on enhanced reaction along a transverse mixing zone. Advances in Water Resources 33(4):525-535.

Willingham TW, Werth CJ, Valocchi AJ. 2008. Evaluation of the effects of porous media structure on mixing-controlled reactions using pore-scale modeling and micromodel experiments. Environmental Science & Technology 42(9):3185-3193.

Yang Y, McCarty PL. 2000. Biologically enhanced dissolution of tetrachloroethene DNAPL. Environmental Science & Technology 34(14):2979-2984.

Yang Y, McCarty PL. 2002. Comparison between donor substrates for biologically enhanced tetrachloroethene DNAPL dissolution. Environmental Science & Technology 36(15):3400-3404.

Appendix A: Numerical Code

```
disp(' ')
disp('Be sure to save and close the Excel spreadsheets')
disp('when you are finished.')

winopen('Model_Inputs/ScenarioInputs.xlsx')
winopen('Model_Inputs/DiscretizationInputs.xlsx')
winopen('Model_Inputs/BioInputs.xlsx')
```

```
function initializesimulation

clear
clc

Nchem = 7;
Nbio = 3;

addpath('Subfunctions\')

%% Import physical parameters and initial conditions from file.

importfile('Model_Inputs\ScenarioInputs.xlsx')

% xL = data(1,1);           % cm
% yL = data(2,1);           % cm
thickness = data(3,1);      % cm
n = data(4,1);              % unitless
flowrate = data(5,1);       % mL/hr (cm^3/hr)
alphaL = data(6,1);         % cm
alphaT = data(7,1);         % cm
tortuosity = data(8,1);     % unitless
K0 = data(9,1);             % cm/hr
Kmin = data(10,1);          % cm/hr
hL = data(11,1);            % cm
Kmod = data(12,1);          % choice (1 or 2)
NAPLmod = zeros(1,Nchem);
NAPLmod(3) = data(13,1);    % choice (1 or 2)
k_NAPL = data(14,1);        % cm/hr

S0_ACE = data(1,5);         % umol/L
S0_H2 = data(2,5);          % umol/L
A0_PCE = data(3,5);         % umol/L
A0_TCE = data(4,5);         % umol/L
A0_DCE = data(5,5);         % umol/L
A0_VC = data(6,5);          % umol/L
A0_ETH = data(7,5);         % umol/L

di_ACE = data(10,5)*10^4*3.6*10^3;  % Converting to cm^2/hr.
di_H2 = data(11,5)*10^4*3.6*10^3;   % Converting to cm^2/hr.
di_PCE = data(12,5)*10^4*3.6*10^3;  % Converting to cm^2/hr.
di_TCE = data(13,5)*10^4*3.6*10^3;  % Converting to cm^2/hr.
di_DCE = data(14,5)*10^4*3.6*10^3;  % Converting to cm^2/hr.
di_VC = data(15,5)*10^4*3.6*10^3;   % Converting to cm^2/hr.
di_ETH = data(16,5)*10^4*3.6*10^3;  % Converting to cm^2/hr.

Asat_PCE = data(18,5);      % umol/L

T = data(7,9);              % hr
delt = data(8,9);           % hr
maxADdeltA = data(9,9);
reportInLvl = data(10,9);   % hr

Xinit = data(12,9);         % mgVSS/L
biorxn = data(13,9);        % choice (1, 2, or 3)
```

```
clear data textdata colheaders

% Input Biological parameters
[parameters_Meth,parameters_195,parameters_BB1,parameters_Dhb,Xf_195,Xf_Meth,Xf_BB1,Xf_Dhb] =↙
inputs;

% Input discretization
importfile1('Model_Inputs\DiscretizationInputs.xlsx')
delx0 = deltaX(~isnan(deltaX));
dely0 = deltaY(~isnan(deltaY));
xL = xL(~isnan(xL));
yL = yL(~isnan(yL));

% Define x- and y-values vectors.
Nx = length(delx0)+1;
delx0(Nx) = delx0(Nx-1);
Ny = length(dely0);
N = Nx*Ny;

delx = zeros(N,1);
dely = zeros(N,1);

for col = 1:Nx
    for row = 1:Ny
        i = (col-1)*Ny+row;
        delx(i) = delx0(col);
        dely(i) = dely0(row);
    end
end

x_values = zeros(Nx,1);
y_values = zeros(Ny,1);

x_values(1) = 0.5*delx0(1);
for i = 2:Nx
    x_values(i) = sum(delx0(1:i-1))+0.5*delx0(i);
end

y_values(1) = 0.5*dely0(1);
for i = 2:Ny
    y_values(i) = sum(dely0(1:i-1))+0.5*dely0(i);
end

% Assume uniform initial chemical distribution.
Sinit_ACE = S0_ACE*ones(N,1);
Sinit_H2 = S0_H2*ones(N,1);
Ainit_PCE = A0_PCE*ones(N,1);
Ainit_TCE = A0_TCE*ones(N,1);
Ainit_DCE = A0_DCE*ones(N,1);
Ainit_VC = A0_VC*ones(N,1);
Ainit_ETH = A0_ETH*ones(N,1);

% Assume initial biomass of 0.15 mg VSS/L
switch biorxn
    case 0
        biotic = 2;
```

```
    Xinit_195 = zeros(N,1);
    Xinit_Meth = zeros(N,1);
    Xinit_BB1 = zeros(N,1);
    Xinit_Dhb = zeros(N,1);

case 1
    biotic = 1;
    Xinit_195 = Xinit*ones(N,1);
    Xinit_Meth = zeros(N,1);
    Xinit_BB1 = zeros(N,1);
    Xinit_Dhb = zeros(N,1);
    cd Bioreactions
    reaction_sys = @reaction_sys_Dhc;
    cd ..
case 2
    biotic = 1;
    Xinit_195 = zeros(N,1);
    Xinit_Meth = Xinit*ones(N,1);
    Xinit_BB1 = zeros(N,1);
    Xinit_Dhb = zeros(N,1);
    cd Bioreactions
    reaction_sys = @reaction_sys_Meth;
    cd ..

case 3
    biotic = 1;
    Xinit_195 = zeros(N,1);
    Xinit_Meth = zeros(N,1);
    Xinit_BB1 = Xinit*ones(N,1);
    Xinit_Dhb = zeros(N,1);
    cd Bioreactions
    reaction_sys = @reaction_sys_BB1;
    cd ..

case 4
    biotic = 1;
    Xinit_195 = Xinit*ones(N,1);
    Xinit_Meth = Xinit*ones(N,1);
    Xinit_BB1 = zeros(N,1);
    Xinit_Dhb = zeros(N,1);
    cd Bioreactions
    reaction_sys = @reaction_sys_Dhc_Meth;
    cd ..

case 5
    biotic = 1;
    Xinit_195 = Xinit*ones(N,1);
    Xinit_Meth = zeros(N,1);
    Xinit_BB1 = Xinit*ones(N,1);
    Xinit_Dhb = zeros(N,1);
    cd Bioreactions
    reaction_sys = @reaction_sys_BB1_Dhc;
    cd ..

case 6
    biotic = 1;
```

```
        Xinit_195 = zeros(N,1);
        Xinit_Meth = Xinit*ones(N,1);
        Xinit_BB1 = Xinit*ones(N,1);
        Xinit_Dhb = zeros(N,1);
        cd Bioreactions
        reaction_sys = @reaction_sys_BB1_Meth;
        cd ..

    case 7
        biotic = 1;
        Xinit_195 = Xinit*ones(N,1);
        Xinit_Meth = Xinit*ones(N,1);
        Xinit_BB1 = Xinit*ones(N,1);
        Xinit_Dhb = zeros(N,1);
        cd Bioreactions
        reaction_sys = @reaction_sys_BB1_Dhc_Meth;
        cd ..

    otherwise
        disp('Error in Bio Reaction options')
        return
end

% Calculate influent specific discharge
qx0 = flowrate/(thickness*yL);

% Solution Parameters for Crank-Nicholson.
theta = 0.5;
tol = 1e-014;
maxit = 60;

% Calculate internally used parameters.
Nt = T/delt;
nADSol = max([1 ceil(delt/2/maxADdeltA)]);
deltA = 0.5*delt/nADSol;
negcheck2track = zeros(2,1);

% Initialize empty vectors for flow field calculations.
K = zeros(N,1);
Knew = zeros(N,1);

%% Initialize variables for the simulation.
S_ACE = Sinit_ACE;
S_H2 = Sinit_H2;
A_PCE = Ainit_PCE;
A_TCE = Ainit_TCE;
A_DCE = Ainit_DCE;
A_VC = Ainit_VC;
A_ETH = Ainit_ETH;

X_195 = Xinit_195;
X_Meth = Xinit_Meth;
X_BB1 = Xinit_BB1;
X_Dhb = Xinit_Dhb;

% b_cells contains the b-vectors in the order of ACE H2 PCE TCE DCE VC ETH
```

```
b = cell(1,Nchem);
parfor i = 1:Nchem
    b{i} = zeros(N,1);
end

%% Calculate the initial flow pattern.

Biomassmat = [X_195 X_Meth X_BB1 X_Dhb];
nXf = n./[Xf_195;Xf_Meth;Xf_BB1;Xf_Dhb];

% The hydraulic conductivity is modeled as in Chu et. al, (2003)
nb = Biomassmat*nXf;

switch Kmod
    case 1
        % Model as a gradually increasing biomass.
        K = K0.*(1-nb/n).^2;

    case 2
        % Model as plugs.
        nbidx = nb<=(Kmin./(nb.*K0));
        K(nbidx) = K0;
        K(~nbidx) = n*Kmin./nb(~nbidx);

    otherwise
        disp('Error in modeling K.  Check the spreadsheet')
        return
end

% Calculate the hydraulic potential at each location
[h,vx,vy] = flownet_2D_gradualy(xL,yL,Nx,Ny,delx,dely,K,qx0,hL,n,nb);

% Calculate the values of the dispersion coefficents.
Dx_ACE = zeros(size(vx));
Dx_H2 = zeros(size(vx));
Dx_PCE = zeros(size(vx));
Dx_TCE = zeros(size(vx));
Dx_DCE = zeros(size(vx));
Dx_VC = zeros(size(vx));
Dx_ETH = zeros(size(vx));

Dy_ACE = zeros(size(vy));
Dy_H2 = zeros(size(vy));
Dy_PCE = zeros(size(vy));
Dy_TCE = zeros(size(vy));
Dy_DCE = zeros(size(vy));
Dy_VC = zeros(size(vy));
Dy_ETH = zeros(size(vy));

% Calculate Dx values for interior locations:
parfor i = Ny+1:N

    col = ceil(i/Ny);

    vyD = (vy(i-Ny+col-1)+vy(i+col)+vy(i-Ny+col-2)+vy(i+col-1))/4;
```

```
    mechDx = (alphaT*vyD^2 + alphaL*vx(i)^2)/sqrt(vyD^2 + vx(i)^2);

    Dx_ACE(i) = mechDx + di_ACE*tortuosity;
    Dx_H2(i) = mechDx + di_H2*tortuosity;
    Dx_PCE(i) = mechDx + di_PCE*tortuosity;
    Dx_TCE(i) = mechDx + di_TCE*tortuosity;
    Dx_DCE(i) = mechDx + di_DCE*tortuosity;
    Dx_VC(i) = mechDx + di_VC*tortuosity;
    Dx_ETH(i) = mechDx + di_ETH*tortuosity;

end

% Calculate the Dx values for inlet locations.
parfor i = 1:Ny

    mechDx = alphaL*vx(i);

    Dx_ACE(i) = mechDx + di_ACE*tortuosity;
    Dx_H2(i) = mechDx + di_H2*tortuosity;
    Dx_PCE(i) = mechDx + di_PCE*tortuosity;
    Dx_TCE(i) = mechDx + di_TCE*tortuosity;
    Dx_DCE(i) = mechDx + di_DCE*tortuosity;
    Dx_VC(i) = mechDx + di_VC*tortuosity;
    Dx_ETH(i) = mechDx + di_ETH*tortuosity;

end

% Dx values at outlet locations are all equal to zero already.

% Calculate internal Dy values.
for col = 1:Nx

    for row = 2:Ny

        i = (col-1)*(Ny+1)+row;

        vxD = (vx(i-col) + vx(i-col+1) + vx(i-col+Ny+1) + vx(i-col+Ny))/4;

        mechDy = (alphaT*vxD^2 + alphaL*vy(i)^2)/sqrt(vxD^2 + vy(i)^2);

        Dy_ACE(i) = mechDy + di_ACE*tortuosity;
        Dy_H2(i) = mechDy + di_H2*tortuosity;
        Dy_PCE(i) = mechDy + di_PCE*tortuosity;
        Dy_TCE(i) = mechDy + di_TCE*tortuosity;
        Dy_DCE(i) = mechDy + di_DCE*tortuosity;
        Dy_VC(i) = mechDy + di_VC*tortuosity;
        Dy_ETH(i) = mechDy + di_ETH*tortuosity;

    end
end

% Dy values at the wall and NAPL Boundary for non-PCE are zero.
% Calculate the Dy value at NAPL Boundary for PCE.
for col = 1:Nx

    i = (col-1)*(Ny+1)+1;
```

```
    vxD = (vx(i-col+1) + vx(i-col+Ny+1))/2;

    mechDy = (alphaT*vxD^2 + alphaL*vy(i)^2)/sqrt(vxD^2 + vy(i)^2);

    Dy_PCE(i) = mechDy + di_PCE*tortuosity;

end

% Initialize vectors and matrices for dissolution and efflux tracking.
timevector = delt*transpose(1:Nt);
PCEdissolutionflux = zeros(1,Nx-1);

PCEdissolutiontrack = zeros(Nt,nADSol*2);
ACEeffluxtrack = zeros(Nt,nADSol*2);
H2effluxtrack = zeros(Nt,nADSol*2);
PCEeffluxtrack = zeros(Nt,nADSol*2);
TCEeffluxtrack = zeros(Nt,nADSol*2);
DCEeffluxtrack = zeros(Nt,nADSol*2);
VCeffluxtrack = zeros(Nt,nADSol*2);
ETHeffluxtrack = zeros(Nt,nADSol*2);

ACEefflux1 = zeros(Ny,1);
H2efflux1 = zeros(Ny,1);
PCEefflux1 = zeros(Ny,1);
TCEefflux1 = zeros(Ny,1);
DCEefflux1 = zeros(Ny,1);
VCefflux1 = zeros(Ny,1);
ETHefflux1 = zeros(Ny,1);

ACEefflux2 = zeros(Ny,1);
H2efflux2 = zeros(Ny,1);
PCEefflux2 = zeros(Ny,1);
TCEefflux2 = zeros(Ny,1);
DCEefflux2 = zeros(Ny,1);
VCefflux2 = zeros(Ny,1);
ETHefflux2 = zeros(Ny,1);

simtime = 0;

save('Model_Inputs\Initialization.mat')
end
```

```
clear
clc

%% matlabpool settings
matlabpool close force
matlabpool 2

%% Operation of the model
load('Model_Inputs/Initialization.mat');

Nchem = 7;
Nbio = 4;

starttime = tic;
addpath('Bioreactions/')
addpath('Subfunctions/')

% Create folder system for outputs.
currenttime = clock;
date = datestr(currenttime,'yyyy.mm.dd-HH.MM');

if ~isdir('Results')
    mkdir('Results');
end

cd('Results')
Mainfolder = [date '_Simulation'];
Concentrationfolder = [date '_Simulation/Concentrations'];
Biomassfolder = [date '_Simulation/Biomass'];
Workspacefolder = [date '_Simulation/Workspace'];
mkdir(Concentrationfolder);
mkdir(Biomassfolder);
mkdir(Workspacefolder);
save([Mainfolder '/Initialization.mat'])
cd('..')

% Establish the simulation time at the beginning in case the simulation is
% a continuation.
simtime0 = simtime;
Nt = (T - simtime0)/delt;

% Set up cell arrays for use in parallel calculations.
Dx = cell(1,Nchem);
Dx{1} = Dx_ACE;
Dx{2} = Dx_H2;
Dx{3} = Dx_PCE;
Dx{4} = Dx_TCE;
Dx{5} = Dx_DCE;
Dx{6} = Dx_VC;
Dx{7} = Dx_ETH;

Dy = cell(1,Nchem);
Dy{1} = Dy_ACE;
Dy{2} = Dy_H2;
Dy{3} = Dy_PCE;
Dy{4} = Dy_TCE;
```

```
Dy{5} = Dy_DCE;
Dy{6} = Dy_VC;
Dy{7} = Dy_ETH;

efflux = cell(1,Nchem);
efflux1 = {ACEefflux1,H2efflux1,PCEefflux1,TCEefflux1,DCEefflux1,VCefflux1,ETHefflux1};
efflux2 = {ACEefflux2,H2efflux2,PCEefflux2,TCEefflux2,DCEefflux2,VCefflux2,ETHefflux2};

effluxtrack_swap = zeros(Nchem,nADSol);
PCEdissolutiontrack_swap = zeros(Nchem,nADSol);


%% Time loop to simulate the experiment.
reformCoeff = 1;
for z = 1:Nt
    %% Form coefficient matrix for AD using Crank-Nicholson for dS/dt
    if reformCoeff >= 1

        disp('whatup?')

        % Store Coeff Matrices in a cell structure for a later parfor loop.
        Coeff = cell(1,Nchem);
        L = cell(1,Nchem);
        U = cell(1,Nchem);

        parfor eqnumber = 1:Nchem
            Coeff{eqnumber} = formCoeff2(theta,deltA,Dx{eqnumber},Dy{eqnumber},vx,vy,Nx,Ny,delx,↙
dely,k_NAPL,NAPLmod(eqnumber));
            [L{eqnumber},U{eqnumber}] = ilu(Coeff{eqnumber},struct('type','ilutp','droptol',1e-6));
        end

        reformCoeff = 0;

        disp('Coefficient Matrices formed')

    end

    %% Store previous values of the concentrations.
    Sold_ACE = S_ACE;
    Sold_H2 = S_H2;
    Aold_PCE = A_PCE;
    Aold_TCE = A_TCE;
    Aold_DCE = A_DCE;
    Aold_VC = A_VC;
    Aold_ETH = A_ETH;

    Xold_195 = X_195;
    Xold_Meth = X_Meth;
    Xold_BB1 = X_BB1;
    Xold_Dhb = X_Dhb;

    %% Build and Solve A-D equations for the first half-step.
    tic

    % Transfer values into the cell structure for parallelization.
    Sstar = cell(1,Nchem);
```

```
S0 = [S0_ACE S0_H2 A0_PCE A0_TCE A0_DCE A0_VC A0_ETH];
Sstar{1} = Sold_ACE;
Sstar{2} = Sold_H2;
Sstar{3} = Aold_PCE;
Sstar{4} = Aold_TCE;
Sstar{5} = Aold_DCE;
Sstar{6} = Aold_VC;
Sstar{7} = Aold_ETH;

Dx = cell(1,Nchem);
Dx{1} = Dx_ACE;
Dx{2} = Dx_H2;
Dx{3} = Dx_PCE;
Dx{4} = Dx_TCE;
Dx{5} = Dx_DCE;
Dx{6} = Dx_VC;
Dx{7} = Dx_ETH;

Dy = cell(1,Nchem);
Dy{1} = Dy_ACE;
Dy{2} = Dy_H2;
Dy{3} = Dy_PCE;
Dy{4} = Dy_TCE;
Dy{5} = Dy_DCE;
Dy{6} = Dy_VC;
Dy{7} = Dy_ETH;

parfor eqnumber = 1:Nchem

    efflux1_temp = efflux1{eqnumber};
    efflux2_temp = efflux2{eqnumber};
    Dy_temp = Dy{eqnumber};

    for i = 1:nADSol

        PCEdissolutionflux_temp = 0*zeros(Nx-1,1);
        Sstar_temp = Sstar{eqnumber};

        if eqnumber == 3

            switch NAPLmod(eqnumber)

                case {1,5}
                    % Local equilibrium
                    for col = 11:(Nx-1)
                        cvn = (col-1)*Ny+1;

                        PCEdissolutionflux_temp(col) = -n*Dy_temp(cvn+col-1)/(3*dely(cvn))*...
                            (-Sstar_temp(cvn+1) + 9*Sstar_temp(cvn) - 8*Asat_PCE);
                    end

                case {2,6}
                    % Local non-equilibrium.
                    for col = 11:(Nx-1)
                        cvn = (col-1)*Ny+1;
```

```
                        PCEdissolutionflux_temp(col) = -n*k_NAPL*(Sstar_temp(cvn) - Asat_PCE);
                    end

                case {3,4}
                    % Triage
                    for col = 11:(Nx-1)
                        cvn = (col-1)*Ny+1;

                        PCEdissolutionflux_temp(col) = n*vx(cvn)*0.1*Asat_PCE;
                    end

                otherwise
                    %No chance in the value of PCEdissolutionflux
            end

            PCEdissolutiontrack_swap(eqnumber,i) = thickness*dot(PCEdissolutionflux_temp,delx0↙
(1:(Nx-1)));

        end

        for row = 1:Ny
            col = Nx-1;
            cvn = (col-1)*Ny + row;

            efflux1_temp(row) = n*vx(cvn+Ny)*Sstar_temp(cvn);
        end

        efflux1{eqnumber} = efflux1_temp;

        % Fill in the b-vectors.
        b{eqnumber} = feval(@formRHS_inletmod_triage,Sstar{eqnumber},S0(eqnumber),theta,deltA,Dx↙
{eqnumber},Dy{eqnumber},vx,vy,Nx,Ny,delx,dely,Asat_PCE,k_NAPL,NAPLmod(eqnumber),b{eqnumber});

        % Calculate A-D for the first half of the time step.
        [Sstar{eqnumber},~] = feval(@cgs,Coeff{eqnumber},b{eqnumber},tol,maxit,L{eqnumber},U↙
{eqnumber},Sstar{eqnumber});

        for row = 1:Ny
            col = Nx-1;
            cvn = (col-1)*Ny + row;

            efflux2_temp(row) = n*vx(cvn+Ny)*Sstar_temp(cvn);
        end

        efflux2{eqnumber} = efflux2_temp;

        efflux{eqnumber} = (efflux1{eqnumber} + efflux2{eqnumber})/2;

        effluxtrack_swap(eqnumber,i) = thickness*dot(efflux{eqnumber},dely0);
    end

end

% Transfer temporary efflux tracking values to the persistent matrices.
% Units in all are (umol*cm^3)/(hr*L)
ACEeffluxtrack(z+(simtime0/delt),(1:nADSol)) = effluxtrack_swap(1,:);
```

```
    H2effluxtrack(z+(simtime0/delt),(1:nADSol)) = effluxtrack_swap(2,:);
    PCEeffluxtrack(z+(simtime0/delt),(1:nADSol)) = effluxtrack_swap(3,:);
    TCEeffluxtrack(z+(simtime0/delt),(1:nADSol)) = effluxtrack_swap(4,:);
    DCEeffluxtrack(z+(simtime0/delt),(1:nADSol)) = effluxtrack_swap(5,:);
    VCeffluxtrack(z+(simtime0/delt),(1:nADSol)) = effluxtrack_swap(6,:);
    ETHeffluxtrack(z+(simtime0/delt),(1:nADSol)) = effluxtrack_swap(7,:);

    PCEdissolutiontrack(z+(simtime0/delt),(1:nADSol)) = PCEdissolutiontrack_swap(3,:);

    toc

    %% Calculate the reactions and update the star2 vectors.
    % Substitute into the beginning of the reaction module.
    switch biotic
        case 1

            disp(reaction_sys)

            % Transfer concentrations to matrix form for use in Bio module.
            ConcMat_old = [cell2mat(Sstar) Xold_195 Xold_Meth Xold_BB1 Xold_Dhb];

            % Check and report if any concentrations are negative before bio
            negcheck1 = min(min(ConcMat_old));

            if negcheck1 < 0
                disp(' ')
                disp(['negcheck1 = ' num2str(negcheck1)])
                disp(' ')
            end

            tic
            ConcMat = Biomodule45_parfor(reaction_sys,delt,ConcMat_old,parameters_195,↙
parameters_Meth,parameters_BB1,parameters_Dhb);
            toc


            % Transfer concentrations to the cell structure for
            % parallelization.
            Sstar2 = mat2cell(ConcMat(:,1:Nchem),N,ones(1,Nchem));
            X_195 = ConcMat(:,Nchem+1);
            X_Meth = ConcMat(:,Nchem+2);
            X_BB1 = ConcMat(:,Nchem+3);
            X_Dhb = ConcMat(:,Nchem+4);

            % Check and record if any concentrations are negative after bio
            negcheck2 = min(min(ConcMat));

            if negcheck2 < 0
                [~,sizenegcheck2track] = size(negcheck2track);
                negcheck2track(:,sizenegcheck2track+1) = [z*delt;negcheck2];
            end

            % Check if H2 is lower than the lowest thermodynamic threshold
            negcheck3 = min(Sstar2{2})-parameters_195(12);

            if negcheck3 < 0
```

```
            disp(['negcheck3 = ' num2str(negcheck3)])
        end

    case 2

        Sstar2 = Sstar;

    otherwise
        disp('error in choosing biotic or not')
        return

end

%% Build and Solve A-D equations for the first half-step.
tic

Sstar = Sstar2;

parfor eqnumber = 1:Nchem

    efflux1_temp = efflux1{eqnumber};
    efflux2_temp = efflux2{eqnumber};
    Dy_temp = Dy{eqnumber};

    for i = 1:nADSol

        PCEdissolutionflux_temp = 0*zeros(Nx-1,1);
        Sstar_temp = Sstar{eqnumber};

        if eqnumber == 3

            switch NAPLmod(eqnumber)

                case {1,5}
                    % Local equilibrium
                    for col = 11:(Nx-1)
                        cvn = (col-1)*Ny+1;

                        PCEdissolutionflux_temp(col) = -n*Dy_temp(cvn+col-1)/(3*dely(cvn))*...
                            (-Sstar_temp(cvn+1) + 9*Sstar_temp(cvn) - 8*Asat_PCE);
                    end

                case {2,6}
                    % Local non-equilibrium.
                    for col = 11:(Nx-1)
                        cvn = (col-1)*Ny+1;

                        PCEdissolutionflux_temp(col) = -n*k_NAPL*(Sstar_temp(cvn) - Asat_PCE);
                    end

                case {3,4}
                    % Triage
                    for col = 11:(Nx-1)
                        cvn = (col-1)*Ny+1;

                        PCEdissolutionflux_temp(col) = n*vx(cvn)*0.1*Asat_PCE;
```

```
                        end

                    otherwise
                        %No chance in the value of PCEdissolutionflux
                end

                PCEdissolutiontrack_swap(eqnumber,i) = thickness*dot(PCEdissolutionflux_temp,delx0↙
(1:(Nx-1)));

            end

            for row = 1:Ny
                col = Nx-1;
                cvn = (col-1)*Ny + row;

                efflux1_temp(row) = n*vx(cvn+Ny)*Sstar_temp(cvn);
            end

            efflux1{eqnumber} = efflux1_temp;

            % Fill in the b-vectors.
            b{eqnumber} = feval(@formRHS_inletmod_triage,Sstar{eqnumber},S0(eqnumber),theta,deltA,Dx↙
{eqnumber},Dy{eqnumber},vx,vy,Nx,Ny,delx,dely,Asat_PCE,k_NAPL,NAPLmod(eqnumber),b{eqnumber});

            % Calculate A-D for the first half of the time step.
            [Sstar{eqnumber},~] = feval(@cgs,Coeff{eqnumber},b{eqnumber},tol,maxit,L{eqnumber},U↙
{eqnumber},Sstar{eqnumber});

            for row = 1:Ny
                col = Nx-1;
                cvn = (col-1)*Ny + row;

                efflux2_temp(row) = n*vx(cvn+Ny)*Sstar_temp(cvn);
            end

            efflux2{eqnumber} = efflux2_temp;

            efflux{eqnumber} = (efflux1{eqnumber} + efflux2{eqnumber})/2;

            effluxtrack_swap(eqnumber,i) = thickness*dot(efflux{eqnumber},dely0);
        end

    end

    % Transfer temporary efflux tracking values to the persistent matrices.
    % Units in all are (umol*cm^3)/(hr*L)
    ACEeffluxtrack(z+(simtime0/delt),(nADSol+1:2*nADSol)) = effluxtrack_swap(1,:);
    H2effluxtrack(z+(simtime0/delt),(nADSol+1:2*nADSol)) = effluxtrack_swap(2,:);
    PCEeffluxtrack(z+(simtime0/delt),(nADSol+1:2*nADSol)) = effluxtrack_swap(3,:);
    TCEeffluxtrack(z+(simtime0/delt),(nADSol+1:2*nADSol)) = effluxtrack_swap(4,:);
    DCEeffluxtrack(z+(simtime0/delt),(nADSol+1:2*nADSol)) = effluxtrack_swap(5,:);
    VCeffluxtrack(z+(simtime0/delt),(nADSol+1:2*nADSol)) = effluxtrack_swap(6,:);
    ETHeffluxtrack(z+(simtime0/delt),(nADSol+1:2*nADSol)) = effluxtrack_swap(7,:);

    PCEdissolutiontrack(z+(simtime0/delt),(nADSol+1:2*nADSol)) = PCEdissolutiontrack_swap(3,:);
```

```
    Sstar_ACE = Sstar{1};
    Sstar_H2 = Sstar{2};
    Astar_PCE = Sstar{3};
    Astar_TCE = Sstar{4};
    Astar_DCE = Sstar{5};
    Astar_VC = Sstar{6};
    Astar_ETH = Sstar{7};

    toc

    S_ACE = Sstar_ACE;
    S_H2 = Sstar_H2;
    A_PCE = Astar_PCE;
    A_TCE = Astar_TCE;
    A_DCE = Astar_DCE;
    A_VC = Astar_VC;
    A_ETH = Astar_ETH;

    %% Calculate dissolution and dechlorination rates
    tic
    dechlor_PCE_195 = dechlorination(X_195,A_PCE,S_H2,parameters_195(1),parameters_195(6),↙
parameters_195(10),parameters_195(12));
    dechlor_TCE_195 = dechlorination(X_195,A_TCE,S_H2,parameters_195(2),parameters_195(7),↙
parameters_195(10),parameters_195(12));
    dechlor_DCE_195 = dechlorination(X_195,A_DCE,S_H2,parameters_195(3),parameters_195(8),↙
parameters_195(10),parameters_195(12));
    dechlor_VC_195 = dechlorination(X_195,A_VC,S_H2,parameters_195(4),parameters_195(9),↙
parameters_195(10),parameters_195(12));

    dechlor_PCE_BB1 = dechlorination(X_BB1,A_PCE,S_ACE,parameters_BB1(1),parameters_BB1(4),↙
parameters_BB1(6),parameters_BB1(8));
    dechlor_TCE_BB1 = dechlorination(X_BB1,A_TCE,S_ACE,parameters_BB1(2),parameters_BB1(5),↙
parameters_BB1(6),parameters_BB1(8));

    toc

    %% Record values to HDD each report interval.

    simtime = timevector(z+simtime0/delt);

    if fix(simtime/reportintvl) == simtime/reportintvl
        csvwrite(['Results/' Concentrationfolder '/S_ACE_' num2str(simtime) '.csv'],S_ACE);
        csvwrite(['Results/' Concentrationfolder '/S_H2_' num2str(simtime) '.csv'],S_H2);
        csvwrite(['Results/' Concentrationfolder '/A_PCE_' num2str(simtime) '.csv'],A_PCE);
        csvwrite(['Results/' Concentrationfolder '/A_TCE_' num2str(simtime) '.csv'],A_TCE);
        csvwrite(['Results/' Concentrationfolder '/A_DCE_' num2str(simtime) '.csv'],A_DCE);
        csvwrite(['Results/' Concentrationfolder '/A_VC_' num2str(simtime) '.csv'],A_VC);
        csvwrite(['Results/' Concentrationfolder '/A_ETH_' num2str(simtime) '.csv'],A_ETH);

        csvwrite(['Results/' Biomassfolder '/X_195_' num2str(simtime) '.csv'],X_195);
        csvwrite(['Results/' Biomassfolder '/X_Meth_' num2str(simtime) '.csv'],X_Meth);
        csvwrite(['Results/' Biomassfolder '/X_BB1_' num2str(simtime) '.csv'],X_BB1);
        csvwrite(['Results/' Biomassfolder '/X_Dhb_' num2str(simtime) '.csv'],X_Dhb);

        save(['Results/' Workspacefolder '/' num2str(simtime) '_hours.mat']);
```

```
    disp(' ')
    disp('Values Recorded')
end



disp(['End of ' num2str(simtime) ' Hours.'])
disp('--------------------------------')
disp(' ')

%% Test for PCE Steady State.
if biotic == 2

    sstest = max(abs(bsxfun(@rdivide,A_PCE,Aold_PCE)-1));

    if  sstest<1e-4
        disp(['Steady state reached at t = ' num2str(z*delt)])
        break
    end
end


%% Test for changed hydraulic conductivity.
% The hydraulic conductivity is modeled as in Chu et. al, (2003)
Biomassmat = [X_195 X_Meth X_BB1 X_Dhb];

nb = Biomassmat*nXf;

switch Kmod
    case 1
        % Model as a gradually increasing biomass.
        Knew = K0.*(1-nb/n).^2;

    case 2
        % Model as plugs.
        nbidx = nb<=(Kmin./(nb.*K0));
        Knew(nbidx) = K0;
        Knew(~nbidx) = n*Kmin./nb(~nbidx);

    otherwise
        disp('Error in modeling K.  Check the spreadsheet')
        return
end

Kchange = abs(bsxfun(@rdivide,Knew,K)-1);

% Set to reform coefficient matrix if significant change.
if max(Kchange) >= 0.005
    reformCoeff = 1;
end

if reformCoeff >= 1
    K = Knew;

    % Calculate the hydraulic potential at each location
    [h,vx,vy] = flownet_2D_gradualy(xL,yL,Nx,Ny,delx,dely,K,qx0,hL,n,nb);
```

```
% Calculate the values of the dispersion coefficents.
Dx_ACE = zeros(size(vx));
Dx_H2 = zeros(size(vx));
Dx_PCE = zeros(size(vx));
Dx_TCE = zeros(size(vx));
Dx_DCE = zeros(size(vx));
Dx_VC = zeros(size(vx));
Dx_ETH = zeros(size(vx));

Dy_H2 = zeros(size(vy));
Dy_PCE = zeros(size(vy));
Dy_TCE = zeros(size(vy));
Dy_DCE = zeros(size(vy));
Dy_VC = zeros(size(vy));
Dy_ETH = zeros(size(vy));

% Calculate Dx values for interior locations:
for i = Ny+1:N

    col = ceil(i/Ny);

    vyD = (vy(i-Ny+col-1)+vy(i-Ny+col-2)+vy(i+col)+vy(i+col-1))/4;

    mechDx = (alphaT*vyD^2 + alphaL*vx(i)^2)/sqrt(vyD^2 + vx(i)^2);

    Dx_ACE(i) = mechDx + di_ACE*tortuosity;
    Dx_H2(i) = mechDx + di_H2*tortuosity;
    Dx_PCE(i) = mechDx + di_PCE*tortuosity;
    Dx_TCE(i) = mechDx + di_TCE*tortuosity;
    Dx_DCE(i) = mechDx + di_DCE*tortuosity;
    Dx_VC(i) = mechDx + di_VC*tortuosity;
    Dx_ETH(i) = mechDx + di_ETH*tortuosity;

end

% Calculate the Dx values for inlet locations.
for i = 1:Ny

    mechDx = alphaL*vx(i);

    Dx_ACE(i) = mechDx + di_ACE*tortuosity;
    Dx_H2(i) = mechDx + di_H2*tortuosity;
    Dx_PCE(i) = mechDx + di_PCE*tortuosity;
    Dx_TCE(i) = mechDx + di_TCE*tortuosity;
    Dx_DCE(i) = mechDx + di_DCE*tortuosity;
    Dx_VC(i) = mechDx + di_VC*tortuosity;
    Dx_ETH(i) = mechDx + di_ETH*tortuosity;

end

% Dx values at outlet locations are all equal to zero already.

% Calculate internal Dy values.
for col = 1:Nx
```

```
            for row = 2:Ny

                i = (col-1)*(Ny+1)+row;

                vxD = (vx(i-col) + vx(i-col+1) + vx(i-col+Ny+1) + vx(i-col+Ny))/4;

                mechDy = (alphaT*vxD^2 + alphaL*vy(i)^2)/sqrt(vxD^2 + vy(i)^2);

                Dy_ACE(i) = mechDy + di_ACE*tortuosity;
                Dy_H2(i) = mechDy + di_H2*tortuosity;
                Dy_PCE(i) = mechDy + di_PCE*tortuosity;
                Dy_TCE(i) = mechDy + di_TCE*tortuosity;
                Dy_DCE(i) = mechDy + di_DCE*tortuosity;
                Dy_VC(i) = mechDy + di_VC*tortuosity;
                Dy_ETH(i) = mechDy + di_ETH*tortuosity;

            end
        end

        % Dy values at the wall and NAPL Boundary for non-PCE are zero.
        % Calculate the Dy value at NAPL Boundary for PCE.
        for col = 1:Nx

            i = (col-1)*(Ny+1)+1;

            vxD = (vx(i-col+1) + vx(i-col+Ny+1))/2;

            mechDy = (alphaT*vxD^2 + alphaL*vy(i)^2)/sqrt(vxD^2 + vy(i)^2);

            Dy_PCE(i) = mechDy + di_PCE*tortuosity;

        end

    end



end

% Create matrices for concentrations for 2-D plotting.
ACEmat = zeros(Ny,Nx);
H2mat = zeros(Ny,Nx);
PCEmat = zeros(Ny,Nx);
TCEmat = zeros(Ny,Nx);
DCEmat = zeros(Ny,Nx);
VCmat = zeros(Ny,Nx);
ETHmat = zeros(Ny,Nx);
X195mat = zeros(Ny,Nx);
XMethmat = zeros(Ny,Nx);

for i = 1:N
    ACEmat(i) = S_ACE(i);
    H2mat(i) = S_H2(i);
    PCEmat(i) = A_PCE(i);
    TCEmat(i) = A_TCE(i);
    DCEmat(i) = A_DCE(i);
```

```
    VCmat(i) = A_VC(i);
    ETHmat(i) = A_ETH(i);
    X195mat(i) = X_195(i);
    XMethmat(i) = X_Meth(i);
end

%% Calculate LE analytical solution if set to LE, Abiotic.
if biotic == 2 && (NAPLmod(3) == 1 || NAPLmod(3) == 5)
    PCEmat_analyt = LEanalytical(xL,vx,delx0,Dy_PCE,x_values,y_values,Asat_PCE);
end


%% Save Results for Later

duration = toc(starttime);
save(['Results/' Mainfolder '/FinalWorkspace.mat'])

beep
```

```
function ConcMat = Biomodule45_parfor(reaction_sys,delt,ConcMat_old,parameters_195,parameters_Meth,↙
parameters_BB1,parameters_Dhb)

% ConcMat contains the concentration vectors in the order of
% [ACE H2 PCE TCE DCE VC ETH X195 XMeth XBB1 XDhb]

nh = 15;

h0 = delt/nh;
reltol = 1e-6;

[N,neq] = size(ConcMat_old);
ConcMat = zeros(size(ConcMat_old));
kMat = zeros(neq,6);
a = [0;1/5;3/10;3/5;1;7/8];
bMat = [
    1/5 3/40    3/10    -11/54  1631/55296
    0   9/40    -9/10   5/2     175/512
    0   0       6/5     -70/27  575/13824
    0   0       0       35/27   44275/110592
    0   0       0       0       253/4096
    0   0       0       0       0
    ];
c4 = [37/378;0;250/621;125/594;0;512/1771];
c5 = [2825/27648;0;18575/48384;13525/55296;277/14336;1/4];

% For each row, solve the coupled Dual Monod equations for these values
% using RKCK.

parfor i = 1:N

    kMat = zeros(neq,6);

    y = transpose(ConcMat_old(i,:));

    % To save time, assume no growth at S < 1.001*Sd and use analytical
    % solution for decay.
    if y(2) < 1.001*parameters_195(12) && y(1) < 1.001*parameters_BB1(8)

        % Assume no growth, decay only.
        y(8) = y(8)*exp(-parameters_195(12)*delt);
        y(9) = y(9)*exp(-parameters_Meth(5)*delt);
        y(10) = y(10)*exp(-parameters_BB1(9)*delt);
        y(11) = y(11)*exp(-parameters_Dhb(9)*delt);

        ConcMat(i,:) = transpose(y);

    else
        % Use adaptive RK-CK method.
        h = h0;
        t = 0;

        while true

            yinit = y;
```

```
            % Calculate k values.
            kMat(:,1) = feval(reaction_sys,yinit,parameters_195,parameters_Meth,parameters_BB1,↙
parameters_Dhb);

            kMat(:,2) = feval(reaction_sys,yinit+kMat*bMat(:,1)*h,parameters_195,parameters_Meth,↙
parameters_BB1,parameters_Dhb);

            kMat(:,3) = feval(reaction_sys,yinit+kMat*bMat(:,2)*h,parameters_195,parameters_Meth,↙
parameters_BB1,parameters_Dhb);

            kMat(:,4) = feval(reaction_sys,yinit+kMat*bMat(:,3)*h,parameters_195,parameters_Meth,↙
parameters_BB1,parameters_Dhb);

            kMat(:,5) = feval(reaction_sys,yinit+kMat*bMat(:,4)*h,parameters_195,parameters_Meth,↙
parameters_BB1,parameters_Dhb);

            kMat(:,6) = feval(reaction_sys,yinit+kMat*bMat(:,5)*h,parameters_195,parameters_Meth,↙
parameters_BB1,parameters_Dhb);

            y4 = yinit + kMat*c4*h;

            y5 = yinit + kMat*c5*h;

            % Check relative error.
            errratio = bsxfun(@rdivide,y4,y5);
            errcheck = max(abs(errratio - 1));

            if errcheck > reltol
                h = h*(reltol/errcheck)^0.25;     % Adjust step size.
                continue
            end

            % Move to next step.
            y = y5;
            t = t+h;

            if t == delt
                break
            end

            % Determine the next value of h, if the job is done.
            hnxt = h*(reltol/errcheck)^0.2;
            if t+hnxt >= delt
                h = delt-t;
                continue
            end

            h = hnxt;

        end

        ConcMat(i,:) = transpose(y);
    end
end
```

```
end
```

```
function [h,vx,vy] = flownet_2D_gradualy(xL,yL,Nx,Ny,delx,dely,K,qx0,hL,n,nb)

% FLOWNET_2D Compute the two-dimensional flow pattern in a porous medium
% with varying hydraulic conductivity.

    % Input variables:
    % xL:       Length of domain in the x-direction (parallel to flow)
    % yL:       Length of domain in the y-direction (transverse to flow)
    % delx:     Vector of delx values, defined in master function.
    % dely:     Vector of dely values, defined in master function.
    % K:        Vector of hydraulic conductivity values defined in master.
    % qx0       Influent Darcy flux
    % hL        Outlet hydraulic potential, constant
    % n         Vector of porosity values.
    % nb        Vector of porosity filled with biomass values.

tic
%% Initialize variables for Simulation.
N = Nx*Ny;
Coeff = spalloc(N,N,5*(Nx-2)*(Ny-2)+4*2*(Nx-2)+4*2*(Ny-2)+3*4);
b = zeros(N,1);
vx = zeros(Ny*(Nx+1),1);
vy = zeros(Nx*(Ny+1),1);

%% Set the coefficient matrix.
% Set the coefficient matrix for the NAPL-inlet corner.
i = 1;
KR = (delx(i)+delx(i+Ny))*K(i)*K(i+Ny)/(K(i)*delx(i+Ny)+K(i+Ny)*delx(i));
KT = (dely(i)+dely(i+1))*K(i)*K(i+1)/(K(i)*dely(i+1)+K(i+1)*dely(i));

Coeff(i,i) = 2*KR*dely(i)/(delx(i)+delx(i+Ny))...
    + 2*KT*delx(i)/(dely(i)+dely(i+1));
Coeff(i,i+Ny) = -2*KR*dely(i)/(delx(i) + delx(i+Ny));
Coeff(i,i+1) = -2*KT*delx(i)/(dely(i) + dely(i+1));

for i = 2:Ny-1

    % Set the coefficient matrix for interior inlet volumes.
    KR = (delx(i)+delx(i+Ny))*K(i)*K(i+Ny)/(K(i)*delx(i+Ny)+K(i+Ny)*delx(i));
    KT = (dely(i)+dely(i+1))*K(i)*K(i+1)/(K(i)*dely(i+1)+K(i+1)*dely(i));
    KB = (dely(i)+dely(i-1))*K(i)*K(i-1)/(K(i)*dely(i-1)+K(i-1)*dely(i));

    Coeff(i,i) = 2*KR*dely(i)/(delx(i)+delx(i+Ny))...
        + 2*KT*delx(i)/(dely(i)+dely(i+1)) + 2*KB*delx(i)/(dely(i)+dely(i-1));
    Coeff(i,i+Ny) = -2*KR*dely(i)/(delx(i) + delx(i+Ny));
    Coeff(i,i+1) = -2*KT*delx(i)/(dely(i) + dely(i+1));
    Coeff(i,i-1) = -2*KB*delx(i)/(dely(i) + dely(i-1));
end

% Set the coefficient matrix for the wall-inlet corner.
i = Ny;
KR = (delx(i)+delx(i+Ny))*K(i)*K(i+Ny)/(K(i)*delx(i+Ny)+K(i+Ny)*delx(i));
KB = (dely(i)+dely(i-1))*K(i)*K(i-1)/(K(i)*dely(i-1)+K(i-1)*dely(i));

Coeff(i,i) = 2*KR*dely(i)/(delx(i)+delx(i+Ny))...
    + 2*KB*delx(i)/(dely(i)+dely(i-1));
```

```
Coeff(i,i+Ny) = -2*KR*dely(i)/(delx(i) + delx(i+Ny));
Coeff(i,i-1) = -2*KB*delx(i)/(dely(i) + dely(i-1));

% Set the coefficient matrix for each column of the interior nodes.
for col = 2:Nx-1

    % Set the coefficient matrix for the NAPL side.
    i = (col-1)*Ny + 1;
    KR = (delx(i)+delx(i+Ny))*K(i)*K(i+Ny)/(K(i)*delx(i+Ny)+K(i+Ny)*delx(i));
    KL = (delx(i)+delx(i-Ny))*K(i)*K(i-Ny)/(K(i)*delx(i-Ny)+K(i-Ny)*delx(i));
    KT = (dely(i)+dely(i+1))*K(i)*K(i+1)/(K(i)*dely(i+1)+K(i+1)*dely(i));

    Coeff(i,i) = 2*KR*dely(i)/(delx(i)+delx(i+Ny))...
        + 2*KL*dely(i)/(delx(i)+delx(i-Ny))...
        + 2*KT*delx(i)/(dely(i)+dely(i+1));
    Coeff(i,i+Ny) = -2*KR*dely(i)/(delx(i) + delx(i+Ny));
    Coeff(i,i-Ny) = -2*KL*dely(i)/(delx(i) + delx(i-Ny));
    Coeff(i,i+1) = -2*KT*delx(i)/(dely(i) + dely(i+1));

    for row = 2:Ny-1

        % Set the coefficient matrix for fully interior nodes.
        i = (col-1)*Ny + row;
        KR = (delx(i)+delx(i+Ny))*K(i)*K(i+Ny)/(K(i)*delx(i+Ny)+K(i+Ny)*delx(i));
        KL = (delx(i)+delx(i-Ny))*K(i)*K(i-Ny)/(K(i)*delx(i-Ny)+K(i-Ny)*delx(i));
        KT = (dely(i)+dely(i+1))*K(i)*K(i+1)/(K(i)*dely(i+1)+K(i+1)*dely(i));
        KB = (dely(i)+dely(i-1))*K(i)*K(i-1)/(K(i)*dely(i-1)+K(i-1)*dely(i));

        Coeff(i,i) = 2*KR*dely(i)/(delx(i)+delx(i+Ny))...
            + 2*KL*dely(i)/(delx(i)+delx(i-Ny))...
            + 2*KT*delx(i)/(dely(i)+dely(i+1))...
            + 2*KB*delx(i)/(dely(i)+dely(i-1));
        Coeff(i,i+Ny) = -2*KR*dely(i)/(delx(i) + delx(i+Ny));
        Coeff(i,i-Ny) = -2*KL*dely(i)/(delx(i) + delx(i-Ny));
        Coeff(i,i+1) = -2*KT*delx(i)/(dely(i) + dely(i+1));
        Coeff(i,i-1) = -2*KB*delx(i)/(dely(i) + dely(i-1));
    end

    % Set the coefficient matrix for the wall side.
    i = col*Ny;
    KR = (delx(i)+delx(i+Ny))*K(i)*K(i+Ny)/(K(i)*delx(i+Ny)+K(i+Ny)*delx(i));
    KL = (delx(i)+delx(i-Ny))*K(i)*K(i-Ny)/(K(i)*delx(i-Ny)+K(i-Ny)*delx(i));
    KB = (dely(i)+dely(i-1))*K(i)*K(i-1)/(K(i)*dely(i-1)+K(i-1)*dely(i));

    Coeff(i,i) = 2*KR*dely(i)/(delx(i)+delx(i+Ny))...
        + 2*KL*dely(i)/(delx(i)+delx(i-Ny))...
        + 2*KB*delx(i)/(dely(i)+dely(i-1));
    Coeff(i,i+Ny) = -2*KR*dely(i)/(delx(i) + delx(i+Ny));
    Coeff(i,i-Ny) = -2*KL*dely(i)/(delx(i) + delx(i-Ny));
    Coeff(i,i-1) = -2*KB*delx(i)/(dely(i) + dely(i-1));

end

% Set the coefficient matrix for NAPL-outlet corner.
i = (Nx-1)*Ny + 1;
KR = 2*K(i);
```

```
KL = (delx(i)+delx(i-Ny))*K(i)*K(i-Ny)/(K(i)*delx(i-Ny)+K(i-Ny)*delx(i));
KT = (dely(i)+dely(i+1))*K(i)*K(i+1)/(K(i)*dely(i+1)+K(i+1)*dely(i));

Coeff(i,i) = KR*dely(i)/delx(i)...
    + 2*KL*dely(i)/(delx(i)+delx(i-Ny))...
    + 2*KT*delx(i)/(dely(i)+dely(i+1));
Coeff(i,i-Ny) = -2*KL*dely(i)/(delx(i) + delx(i-Ny));
Coeff(i,i+1) = -2*KT*delx(i)/(dely(i) + dely(i+1));

for row = 2:Ny-1

    % Set the coefficient matrix for interior outlet volumes.
    i = (Nx-1)*Ny + row;
    KR = 2*K(i);
    KL = (delx(i)+delx(i-Ny))*K(i)*K(i-Ny)/(K(i)*delx(i-Ny)+K(i-Ny)*delx(i));
    KT = (dely(i)+dely(i+1))*K(i)*K(i+1)/(K(i)*dely(i+1)+K(i+1)*dely(i));
    KB = (dely(i)+dely(i-1))*K(i)*K(i-1)/(K(i)*dely(i-1)+K(i-1)*dely(i));

    Coeff(i,i) = KR*dely(i)/delx(i)...
        + 2*KL*dely(i)/(delx(i)+delx(i-Ny))...
        + 2*KT*delx(i)/(dely(i)+dely(i+1))...
        + 2*KB*delx(i)/(dely(i)+dely(i-1));
    Coeff(i,i-Ny) = -2*KL*dely(i)/(delx(i) + delx(i-Ny));
    Coeff(i,i+1) = -2*KT*delx(i)/(dely(i) + dely(i+1));
    Coeff(i,i-1) = -2*KB*delx(i)/(dely(i) + dely(i-1));
end

%Set the coefficient matrix for wall-outlet corner.
i = N;
KR = 2*K(i);
KL = (delx(i)+delx(i-Ny))*K(i)*K(i-Ny)/(K(i)*delx(i-Ny)+K(i-Ny)*delx(i));
KB = (dely(i)+dely(i-1))*K(i)*K(i-1)/(K(i)*dely(i-1)+K(i-1)*dely(i));

Coeff(i,i) = KR*dely(i)/delx(i)...
    + 2*KL*dely(i)/(delx(i)+delx(i-Ny))...
    + 2*KB*delx(i)/(dely(i)+dely(i-1));
Coeff(i,i-Ny) = -2*KL*dely(i)/(delx(i) + delx(i-Ny));
Coeff(i,i-1) = -2*KB*delx(i)/(dely(i) + dely(i-1));

%% Modify the RHS vector for inlet and outlet boundary conditions.
for i = 1:Ny

    % Inlet condition.
    b(i) = qx0*dely(i);
end

for i = ((Nx-1)*Ny+1):N

    % Outlet condition
    b(i) = 2*K(i)*dely(i)/delx(i)*hL;
end
toc
%% Solve the linear system of equations for pressures.
tic
h = Coeff\b;
toc
```

```
%% Calculate vectors containing the x- and y-velocity values at each CV.
tic
% Calculate the x-velocity vector at interior locations.
for i = Ny+1:N
    KL = (delx(i)+delx(i-Ny))*K(i)*K(i-Ny)/(K(i)*delx(i-Ny)+K(i-Ny)*delx(i));
    n_eq = n - (nb(i)*delx(i-Ny)+nb(i-Ny)*delx(i))/(delx(i) + delx(i-Ny));

    vx(i) = -2*KL*(h(i)-h(i-Ny))/((delx(i)+delx(i-Ny))*n_eq);
end

% Calculate the x-velocity at the inlet locations.
for i = 1:Ny
    vx(i) = qx0/(n-nb(i));
end

% Calculate the x-velocity at the outlet locations.
for i = N+1:N+Ny
    KL = 2*K(i-Ny);
    vx(i) = -KL*(hL-h(i-Ny))/(delx(i-Ny)*(n-nb(i-Ny)));
end

% Calculate the y-velocity vector.  It remains 0 at wall and NAPL
for col = 1:Nx

    for row = 2:Ny

        i = (col-1)*(Ny+1)+row;
        K_eq = (dely(i-col)+dely(i-col+1))*K(i-col)*K(i-col+1)...
            /(dely(i-col+1)*K(i-col)+dely(i-col)*K(i-col+1));
        n_eq = n-(dely(i-col+1)*nb(i-col)+dely(i-col)*nb(i-col+1))...
            /(dely(i-col+1)+dely(i-col));
        vy(i) = -2*K_eq*(h(i-col+1)-h(i-col))/((dely(i-col)+dely(i-col+1))*n_eq);

    end

end
toc
end
```

```
function Coeff = formCoeff2(theta,deltA,Dx,Dy,vx,vy,Nx,Ny,delx,dely,~,~)
N = Nx*Ny;

index1 = zeros(5*N,1);
index2 = zeros(5*N,1);
Values = zeros(5*N,1);

% Fill in the Coefficient Matrix for the interior-NAPL corner.
cvn = 1;
col = 1;

index1(5*(cvn-1)+1:5*(cvn-1)+5) = cvn*ones(5,1);
index2(5*(cvn-1)+1:5*(cvn-1)+5) = [cvn;cvn+1;1;cvn+Ny;1];

% Fixed flux of PCE into the system.
Values(5*(cvn-1)+1) = 1 - theta*deltA*(...
    - 2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
    - vx(cvn+Ny)*delx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
    - 2*Dy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))...
    - dely(cvn+1)*vy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn)));
Values(5*(cvn-1)+2) = -theta*deltA*(...
    2*Dy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))...
    - dely(cvn+1)*vy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn)));
Values(5*(cvn-1)+4) = -theta*deltA*(...
    2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
    - vx(cvn+Ny)*delx(cvn)/((delx(cvn)+delx(cvn+1))*delx(cvn)));

for row = 2:Ny-1

    % Build Coefficient Matrix for interior inlet volumes.
    col = 1;
    cvn = (col-1)*Ny+row;

    index1(5*(cvn-1)+1:5*(cvn-1)+5) = cvn*ones(5,1);
    index2(5*(cvn-1)+1:5*(cvn-1)+5) = [cvn;cvn+1;cvn-1;cvn+Ny;1];

    % Fill in the Coefficient matrix for the inlet volumes.
    Values(5*(cvn-1)+1) = 1 - theta*deltA*(...
        - 2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
        - vx(cvn+Ny)*delx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
        - 2*Dy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))...
        - vy(cvn+col)*dely(cvn+1)/((dely(cvn)+dely(cvn+1))*dely(cvn))...
        - 2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))...
        + vy(cvn+col-1)*dely(cvn-1)/((dely(cvn)+dely(cvn-1))*dely(cvn)));
    Values(5*(cvn-1)+2) = -theta*deltA*(...
        2*Dy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))...
        - vy(cvn+col)*dely(cvn)/((dely(cvn)+dely(cvn+1))*dely(cvn)));
    Values(5*(cvn-1)+3) = -theta*deltA*(...
        2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))...
        + vy(cvn+col-1)*dely(cvn)/((dely(cvn)+dely(cvn-1))*dely(cvn)));
    Values(5*(cvn-1)+4) = -theta*deltA*(...
        2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
        - vx(cvn+Ny)*delx(cvn)/((delx(cvn)+delx(cvn+Ny))*delx(cvn)));

end
```

```
% Fill in Coefficient matrix for Wall-inlet corner.

% Calculate the control volume number.
row = Ny;
col = 1;
cvn = (col-1)*Ny + row;

index1(5*(cvn-1)+1:5*(cvn-1)+5) = cvn*ones(5,1);
index2(5*(cvn-1)+1:5*(cvn-1)+5) = [cvn;1;cvn-1;cvn+Ny;1];

% Fill in the Coefficient matrix for the wall-inlet volume.
Values(5*(cvn-1)+1) = 1 - theta*deltA*(...
    - 2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
    - vx(cvn+Ny)*delx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
    - 2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))...
    + vy(cvn+col-1)*dely(cvn-1)/((dely(cvn)+dely(cvn-1))*dely(cvn)));
Values(5*(cvn-1)+3) = -theta*deltA*(...
    2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))...
    + vy(cvn+col-1)*dely(cvn)/((dely(cvn)+dely(cvn-1))*dely(cvn)));
Values(5*(cvn-1)+4) = -theta*deltA*(...
    2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
    - vx(cvn+Ny)*delx(cvn)/((delx(cvn)+delx(cvn+Ny))*delx(cvn)));

%Calculate values for internal columns.
for col = 2:Nx-1

    % NAPL Side control volume coefficient matrix.
    row = 1;
    cvn = (col-1)*Ny + row;

    % Fill in the coefficient matrix
    index1(5*(cvn-1)+1:5*(cvn-1)+5) = cvn*ones(5,1);
    index2(5*(cvn-1)+1:5*(cvn-1)+5) = [cvn;cvn+1;1;cvn+Ny;cvn-Ny];

    % Non-PCE Matrix
    Values(5*(cvn-1)+1) = 1 - theta*deltA*(...
        - 2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
        - vx(cvn+Ny)*delx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
        - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
        + vx(cvn)*delx(cvn-Ny)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
        - 2*Dy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))...
        - vy(cvn+col)*dely(cvn+1)/((dely(cvn)+dely(cvn))*dely(cvn)));
    Values(5*(cvn-1)+2) = -theta*deltA*(...
        2*Dy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))...
        - vy(cvn+col)*dely(cvn)/((dely(cvn)+dely(cvn+1))*dely(cvn)));
    Values(5*(cvn-1)+4) = -theta*deltA*(...
        2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
        - vx(cvn+Ny)*delx(cvn)/((delx(cvn)+delx(cvn+Ny))*delx(cvn)));
    Values(5*(cvn-1)+5) = -theta*deltA*(...
        2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
        + vx(cvn)*delx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn)));

    % Fill in Coefficient matrix for completely interior nodes.
    for row = 2:Ny-1
        cvn = (col-1)*Ny + row;
```

```
        index1(5*(cvn-1)+1:5*(cvn-1)+5) = cvn*ones(5,1);
        index2(5*(cvn-1)+1:5*(cvn-1)+5) = [cvn;cvn+1;cvn-1;cvn+Ny;cvn-Ny];

        % Fill in the coefficient matrix for this row.
        Values(5*(cvn-1)+1) = 1 - theta*deltA*(...
            - 2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
            - vx(cvn+Ny)*delx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
            - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
            + vx(cvn)*delx(cvn-Ny)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
            - 2*Dy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))...
            - vy(cvn+col)*dely(cvn+1)/((dely(cvn)+dely(cvn+1))*dely(cvn))...
            - 2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))...
            + vy(cvn+col-1)*dely(cvn-1)/((dely(cvn)+dely(cvn-1))*dely(cvn)));
        Values(5*(cvn-1)+2) = -theta*deltA*(...
            2*Dy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))...
            - vy(cvn+col)*dely(cvn)/((dely(cvn)+dely(cvn+1))*dely(cvn)));
        Values(5*(cvn-1)+3) = -theta*deltA*(...
            2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))...
            + vy(cvn+col-1)*dely(cvn)/((dely(cvn)+dely(cvn-1))*dely(cvn)));
        Values(5*(cvn-1)+4) = -theta*deltA*(...
            2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
            - vx(cvn+Ny)*delx(cvn)/((delx(cvn)+delx(cvn+Ny))*delx(cvn)));
        Values(5*(cvn-1)+5) = -theta*deltA*(...
            2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
            + vx(cvn)*delx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn)));
    end

    %Fill in elements at North B.C.
    cvn = col*Ny;

    index1(5*(cvn-1)+1:5*(cvn-1)+5) = cvn*ones(5,1);
    index2(5*(cvn-1)+1:5*(cvn-1)+5) = [cvn;1;cvn-1;cvn+Ny;cvn-Ny];

    % Fill in the Coefficient Matrix
    Values(5*(cvn-1)+1) = 1 - theta*deltA*(...
        - 2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
        - vx(cvn+Ny)*delx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
        - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
        + vx(cvn)*delx(cvn-Ny)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
        - 2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))...
        + vy(cvn+col-1)*dely(cvn-1)/((dely(cvn)+dely(cvn-1))*dely(cvn)));
    Values(5*(cvn-1)+3) = -theta*deltA*(...
        2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))...
        + vy(cvn+col-1)*dely(cvn)/((dely(cvn)+dely(cvn-1))*dely(cvn)));
    Values(5*(cvn-1)+4) = -theta*deltA*(...
        2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))...
        - vx(cvn+Ny)*delx(cvn)/((delx(cvn)+delx(cvn+Ny))*delx(cvn)));
    Values(5*(cvn-1)+5) = -theta*deltA*(...
        2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
        + vx(cvn)*delx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn)));
end

% Calculate the rows for the nodes at the outlet.
% Begin with the NAPL-outlet Corner.
col = Nx;
row = 1;
```

```
cvn = (col-1)*Ny+row;

index1(5*(cvn-1)+1:5*(cvn-1)+5) = cvn*ones(5,1);
index2(5*(cvn-1)+1:5*(cvn-1)+5) = [cvn;1;1;1;cvn-Ny];

% Fill in the coefficient matrix
Values(5*(cvn-1)+1) = 1 - theta*deltA*(...
    - vx(cvn+Ny)/(delx(cvn))...
    - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
    + vx(cvn)*delx(cvn-Ny)/((delx(cvn)+delx(cvn-Ny))*delx(cvn)));
Values(5*(cvn-1)+5) = -theta*deltA*(...
    2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
    + vx(cvn)*delx(cvn-Ny)/((delx(cvn)+delx(cvn-Ny))*delx(cvn)));

% Interior outlet control volumes.
for row = 2:Ny-1
    cvn = (col-1)*Ny + row;

    index1(5*(cvn-1)+1:5*(cvn-1)+5) = cvn*ones(5,1);
    index2(5*(cvn-1)+1:5*(cvn-1)+5) = [cvn;1;1;1;cvn-Ny];

    % Fill in the coefficient matrix for this row.
    Values(5*(cvn-1)+1) = 1 - theta*deltA*(...
        - vx(cvn+Ny)/(delx(cvn))...
        - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
        + vx(cvn)*delx(cvn-Ny)/((delx(cvn)+delx(cvn-Ny))*delx(cvn)));
    Values(5*(cvn-1)+5) = -theta*deltA*(...
        2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
        + vx(cvn)*delx(cvn-Ny)/((delx(cvn)+delx(cvn-Ny))*delx(cvn)));
end

% Fill in coefficient matrix for wall-outlet corner.
col = Nx;
cvn = N;

index1(5*(cvn-1)+1:5*(cvn-1)+5) = cvn*ones(5,1);
index2(5*(cvn-1)+1:5*(cvn-1)+5) = [cvn;1;1;1;cvn-Ny];

% Fill in the coefficient matrix for this row.
Values(5*(cvn-1)+1) = 1 - theta*deltA*(...
    - vx(cvn+Ny)/(delx(cvn))...
    - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
    + vx(cvn)*delx(cvn-Ny)/((delx(cvn)+delx(cvn-Ny))*delx(cvn)));
Values(5*(cvn-1)+5) = -theta*deltA*(...
    2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))...
    + vx(cvn)*delx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn)));

Coeff = sparse(index1,index2,Values);

end
```

```
function b = formRHS_inletmod_triage(Sstar,S0,theta,deltA,Dx,Dy,vx,vy,Nx,Ny,delx,dely,Asat_PCE,↙
k_NAPL,NAPLcond,b)

%  PCEdissolutionflux = zeros(1,Nx);
%
% switch NAPLcond
%     case 0
%         % Nothing happens.
%     case 1
%         % Local equilibrium
%         for col = 11:(Nx-1)
%             cvn = (col-1)*Ny+1;
%
%             PCEdissolutionflux(col) = -Dy(cvn+col-1)/(3*dely(cvn))*...
%                 (-Sstar(cvn+1) + 9*Sstar(cvn) - 8*Asat_PCE);
%         end
%
%     case 2
%         % Local non-equilibrium.
%         for col = 11:(Nx-1)
%             cvn = (col-1)*Ny+1;
%
%             PCEdissolutionflux(col) = -k_NAPL*(Sstar(cvn) - Asat_PCE);
%         end
%
%     case 3
%         % Triage
%         for col = 11:(Nx-1)
%             cvn = (col-1)*Ny+1;
%
%             PCEdissolutionflux(col) = vx(cvn)*0.1*Asat_PCE;
%         end
% end
if NAPLcond == 4 || NAPLcond == 3

    PCEdissolutionflux = zeros(1,Nx-1);

    for col = 11:(Nx-1)
        cvn = (col-1)*Ny + 1;
        PCEdissolutionflux(col) = vx(cvn)*0.1*Asat_PCE;
    end
end

if NAPLcond == 5 || NAPLcond == 1

    PCEdissolutionflux = zeros(1,Nx-1);

    % Local equilibrium
    for col = 11:(Nx-1)
        cvn = (col-1)*Ny+1;

        PCEdissolutionflux(col) = -Dy(cvn+col-1)/(3*dely(cvn))*...
            (-Sstar(cvn+1) + 9*Sstar(cvn) - 8*Asat_PCE);
    end
end
```

```
if NAPLcond == 6 || NAPLcond == 2

    PCEdissolutionflux = zeros(1,Nx-1);

    % Local Mass Transfer limitation
    for col = 11:(Nx-1)
        cvn = (col-1)*Ny+1;

        PCEdissolutionflux(col) = -k_NAPL*(Sstar(cvn)-Asat_PCE);
    end
end


% Fill in the b-vector for the Inlet-NAPL corner
cvn = 1;
col = 1;

switch NAPLcond
    case 0
        % Non-PCE vector
        b(cvn) = Sstar(cvn)...
            + S0*deltA/delx(cvn)*vx(cvn)...
            + (1-theta)*deltA*(...
              2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)-Sstar(cvn))...
            - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar(cvn)↙
*delx(cvn+Ny))...
            + 2*Dy(cvn+col)/(dely(cvn)^2)*(Sstar(cvn+1) - Sstar(cvn))...
            - vy(cvn+col)/(2*dely(cvn))*(Sstar(cvn+1)*dely(cvn) + Sstar(cvn)*dely(cvn+1)));

    case 1
        % PCE, Assume Local equilibrium.
        b(cvn) = Sstar(cvn)...
            + S0*deltA/delx(cvn)*vx(cvn)...
            + (1-theta)*deltA*(...
              2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)-Sstar(cvn))...
            - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar(cvn)↙
*delx(cvn+Ny))...
            + 2*Dy(cvn+col)/(dely(cvn)^2)*(Sstar(cvn+1) - Sstar(cvn))...
            - vy(cvn+col)/(2*dely(cvn))*(Sstar(cvn+1)*dely(cvn) + Sstar(cvn)*dely(cvn+1)));

    case 2
        %Assume local non-equilibrium
        b(cvn) = Sstar(cvn)...
            + S0*deltA/delx(cvn)*vx(cvn)...
            + (1-theta)*deltA*(...
              2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)-Sstar(cvn))...
            - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar(cvn)↙
*delx(cvn+Ny))...
            + 2*Dy(cvn+col)/(dely(cvn)^2)*(Sstar(cvn+1) - Sstar(cvn))...
            - vy(cvn+col)/(2*dely(cvn))*(Sstar(cvn+1)*dely(cvn) + Sstar(cvn)*dely(cvn+1)));

    case {3,4,5,6}
        % Mass Balance Triage case
        b(cvn) = Sstar(cvn)...
            + S0*deltA/delx(cvn)*vx(cvn)...
            + (1-theta)*deltA*(...
```

```
              2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)-Sstar(cvn))...
            - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar(cvn)↙
*delx(cvn+Ny))...
            + 2*Dy(cvn+col)/(dely(cvn)^2)*(Sstar(cvn+1) - Sstar(cvn))...
            - vy(cvn+col)/(2*dely(cvn))*(Sstar(cvn+1)*dely(cvn) + Sstar(cvn)*dely(cvn+1)));

    otherwise
        disp('Error in NAPL model')

        return
end

for row = 2:Ny-1

    col = 1;
    cvn = (col-1)*Ny + row;

    % Fill in the b-vectors.
    b(cvn) = Sstar(cvn)...
        + S0*deltA/delx(cvn)*vx(cvn)...
        + (1-theta)*deltA*(...
          2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar(cvn))...
        - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar(cvn)*delx↙
(cvn+Ny))...
        + 2*Dy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))*(Sstar(cvn+1) - Sstar(cvn))...
        - vy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))*(Sstar(cvn+1)*dely(cvn) + Sstar(cvn)*dely↙
(cvn+1))...
        - 2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn) - Sstar(cvn-1))...
        + vy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn)*dely(cvn-1) + Sstar(cvn-1)↙
*dely(cvn)));
end

% Fill in the b-vectors at the wall-inlet corner.
col = 1;
row = Ny;
cvn = (col-1)*Ny+row;

% Fill in the b-vectors.
b(cvn) = Sstar(cvn)...
    + S0*deltA/delx(cvn)*vx(cvn)...
    + (1-theta)*deltA*(...
      2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar(cvn))...
    - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar(cvn)*delx↙
(cvn+Ny))...
    - 2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn) - Sstar(cvn-1))...
    + vy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn)*dely(cvn-1) + Sstar(cvn-1)*dely↙
(cvn)));

for col = 2:10

    % Apply the b-vectors at the NAPL side.
    row = 1;
    cvn = (col-1)*Ny + row;

    b(cvn) = Sstar(cvn)...
        + (1-theta)*deltA*(...
```

```
          2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar(cvn))...
        - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar(cvn)*delx↙
(cvn+Ny))...
        - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn) - Sstar(cvn-Ny))...
        + vx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn)*delx(cvn-Ny) + Sstar(cvn-Ny)*delx↙
(cvn))...
        + Dy(cvn+col)/dely(cvn)^2*(Sstar(cvn+1) - Sstar(cvn))...
        - vy(cvn+col)/(2*dely(cvn))*(Sstar(cvn+1) + Sstar(cvn)));

    % Fill in b-vectors for completely interior nodes.
    for row = 2:Ny-1

        cvn = (col-1)*Ny + row;

        % Fill in the b-vectors.
        b(cvn) = Sstar(cvn) + (1-theta)*deltA*(...
              2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar(cvn))...
            - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar(cvn)↙
*delx(cvn+Ny))...
            - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn) - Sstar(cvn-Ny))...
            + vx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn)*delx(cvn-Ny) + Sstar(cvn-Ny)↙
*delx(cvn))...
            + 2*Dy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))*(Sstar(cvn+1) - Sstar(cvn))...
            - vy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))*(Sstar(cvn+1)*dely(cvn) + Sstar(cvn)↙
*dely(cvn+1))...
            - 2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn) - Sstar(cvn-1))...
            + vy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn)*dely(cvn-1) + Sstar(cvn-↙
1)*dely(cvn-1)));
    end

    % Apply b-vectors at wall side.
    row = Ny;
    cvn = (col-1)*Ny+row;

    b(cvn) = Sstar(cvn) + (1-theta)*deltA*(...
        2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar(cvn))...
        - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar(cvn)*delx↙
(cvn+Ny))...
        - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn) - Sstar(cvn-Ny))...
        + vx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn)*delx(cvn-Ny) + Sstar(cvn-Ny)*delx↙
(cvn))...
        - 2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn) - Sstar(cvn-1))...
        + vy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn)*dely(cvn-1) + Sstar(cvn-1)↙
*dely(cvn)));
end

for col = 11:(Nx-1)

    % Apply the b-vectors at the NAPL side.
    row = 1;
    cvn = (col-1)*Ny + row;

    switch NAPLcond
        case 0
            % Non-PCE vector.
            b(cvn) = Sstar(cvn)...
```

```
                + (1-theta)*deltA*(...
                  2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar(cvn))...
                - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar↙
(cvn)*delx(cvn+Ny))...
                - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn) - Sstar(cvn-Ny))...
                + vx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn)*delx(cvn-Ny) + Sstar(cvn-↙
Ny)*delx(cvn))...
                + Dy(cvn+col)/dely(cvn)^2*(Sstar(cvn+1) - Sstar(cvn))...
                - vy(cvn+col)/(2*dely(cvn))*(Sstar(cvn+1) + Sstar(cvn)));

%         case 1
%             Assume Local equilibrium.
%             b(cvn) = Sstar(cvn)...
%                 + Dy(cvn+col-1)*deltA/dely(cvn)^2*(8/3)*Asat_PCE...
%                 + (1-theta)*deltA*(...
%                   2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar↙
(cvn))...
%                 - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar↙
(cvn)*delx(cvn+Ny))...
%                 - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn) + Sstar(cvn-Ny))...
%                 + vx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn)*delx(cvn-Ny) + Sstar↙
(cvn-Ny)*delx(cvn))...
%                 + Dy(cvn+col)/dely(cvn)^2*(Sstar(cvn+1) - Sstar(cvn))...
%                 - vy(cvn+col)/(2*dely(cvn))*(Sstar(cvn+1) + Sstar(cvn))...
%                 - Dy(cvn+col-1)/dely(cvn)^2*(...
%                 - (1/3)*Sstar(cvn+1) + 3*Sstar(cvn)));
%
%         case 2
%             Assume local non-equilibrium
%             b(cvn) = Sstar(cvn) + k_NAPL*deltA/dely(cvn)*Asat_PCE...
%                 + (1-theta)*deltA*(-k_NAPL/dely(cvn)*Sstar(cvn)...
%                 + 2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar↙
(cvn))...
%                 - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar↙
(cvn)*delx(cvn+Ny))...
%                 - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn) + Sstar(cvn-Ny))...
%                 + vx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn)*delx(cvn-Ny) + Sstar↙
(cvn-Ny)*delx(cvn))...
%                 + Dy(cvn+col)/dely(cvn)^2*(Sstar(cvn+1) - Sstar(cvn))...
%                 - vy(cvn+col)/(2*dely(cvn))*(Sstar(cvn+1) + Sstar(cvn)));
%
%         case 3
%             Mass Balance Triage: fixed PCE flux.
%             b(cvn) = Sstar(cvn)...
%                 + vx(cvn)*0.1*deltA/dely(cvn)*Asat_PCE...
%                 + (1-theta)*deltA*(...
%                   2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar↙
(cvn))...
%                 - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar↙
(cvn)*delx(cvn+Ny))...
%                 - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn) - Sstar(cvn-Ny))...
%                 + vx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn)*delx(cvn-Ny) + Sstar↙
(cvn-Ny)*delx(cvn))...
%                 + Dy(cvn+col)/dely(cvn)^2*(Sstar(cvn+1) - Sstar(cvn))...
%                 - vy(cvn+col)/(2*dely(cvn))*(Sstar(cvn+1) + Sstar(cvn)));
        case {1,2,3,4,5,6}
```

```
            b(cvn) = Sstar(cvn)...
                + deltA/dely(cvn)*PCEdissolutionflux(col)...
                + (1-theta)*deltA*(...
                  2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar(cvn))...
                - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar↙
(cvn)*delx(cvn+Ny))...
                - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn) - Sstar(cvn-Ny))...
                + vx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn)*delx(cvn-Ny) + Sstar(cvn-↙
Ny)*delx(cvn))...
                + Dy(cvn+col)/dely(cvn)^2*(Sstar(cvn+1) - Sstar(cvn))...
                - vy(cvn+col)/(2*dely(cvn))*(Sstar(cvn+1) + Sstar(cvn)));

        otherwise
            disp('Error in NAPL model')
            return
    end

    % Fill in b-vectors for completely interior nodes.
    for row = 2:Ny-1

        cvn = (col-1)*Ny + row;

        % Fill in the b-vectors.
        b(cvn) = Sstar(cvn) + (1-theta)*deltA*(...
              2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar(cvn))...
            - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar(cvn)↙
*delx(cvn+Ny))...
            - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn) - Sstar(cvn-Ny))...
            + vx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn)*delx(cvn-Ny) + Sstar(cvn-Ny)↙
*delx(cvn))...
            + 2*Dy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))*(Sstar(cvn+1) - Sstar(cvn))...
            - vy(cvn+col)/((dely(cvn)+dely(cvn+1))*dely(cvn))*(Sstar(cvn+1)*dely(cvn) + Sstar(cvn)↙
*dely(cvn+1))...
            - 2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn) - Sstar(cvn-1))...
            + vy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn)*dely(cvn-1) + Sstar(cvn-↙
1)*dely(cvn-1));
    end

    % Apply b-vectors at wall side.
    row = Ny;
    cvn = (col-1)*Ny+row;

    b(cvn) = Sstar(cvn) + (1-theta)*deltA*(...
        2*Dx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny) - Sstar(cvn))...
        - vx(cvn+Ny)/((delx(cvn)+delx(cvn+Ny))*delx(cvn))*(Sstar(cvn+Ny)*delx(cvn) + Sstar(cvn)*delx↙
(cvn+Ny))...
        - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn) - Sstar(cvn-Ny))...
        + vx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn)*delx(cvn-Ny) + Sstar(cvn-Ny)*delx↙
(cvn))...
        - 2*Dy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn) - Sstar(cvn-1))...
        + vy(cvn+col-1)/((dely(cvn)+dely(cvn-1))*dely(cvn))*(Sstar(cvn)*dely(cvn-1) + Sstar(cvn-1)↙
*dely(cvn)));
end

% Apply the b-vectors at the outlet.
```

```
% Fill in the outlet control volumes.
col = Nx;

for row = 1:Ny

    % Apply the b-vectors at interior outlet volumes.
    cvn = (col-1)*Ny + row;

    % Fill in the b-vectors.
    b(cvn) = Sstar(cvn) + (1-theta)*deltA*(...
        - vx(cvn+Ny)/delx(cvn)*Sstar(cvn)...
        - 2*Dx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn) - Sstar(cvn-Ny))...
        + vx(cvn)/((delx(cvn)+delx(cvn-Ny))*delx(cvn))*(Sstar(cvn)*delx(cvn-Ny) + Sstar(cvn-Ny)*delx↙
(cvn)));
end
```

```
function dechlorVect = dechlorination(X,A,S,q_max,Ka,Ks,Sd)
% Calculate dechlorination rate of a chlorinated ethene by a specific
% species of microorganism.  All inputs must be vectors of the same size.

dechlorVect = zeros(size(A));

dechlorgo = S>Sd;

dechlorVect(dechlorgo) = q_max.*X(dechlorgo).*A(dechlorgo)./(Ka+A(dechlorgo)).*(S(dechlorgo)-Sd)./↙
(Ks+S(dechlorgo)-Sd);

end
```

```
function importfile(fileToRead1)
%IMPORTFILE(FILETOREAD1)
%  Imports data from the specified file
%  FILETOREAD1:  file to read

%  Auto-generated by MATLAB on 03-Aug-2010 13:20:05

% Import the file
newData1 = importdata(fileToRead1);

% For some XLS and other spreadsheet files, returned data are packed
% within an extra layer of structures.  Unpack them.
fields = fieldnames(newData1.data);
newData1.data = newData1.data.(fields{1});
fields = fieldnames(newData1.textdata);
newData1.textdata = newData1.textdata.(fields{1});

% Create new variables in the base workspace from those fields.
vars = fieldnames(newData1);
for i = 1:length(vars)
    assignin('caller', vars{i}, newData1.(vars{i}));
end
```

```
function importfile1(fileToRead1)
%IMPORTFILE1(FILETOREAD1)
%  Imports data from the specified file
%  FILETOREAD1:  file to read

%  Auto-generated by MATLAB on 23-May-2011 14:00:20

% Import the file
sheetName='Sheet1';
[numbers, strings, raw] = xlsread(fileToRead1, sheetName);
if ~isempty(numbers)
    newData1.data =  numbers;
end

if ~isempty(strings) && ~isempty(numbers)
    [strRows, strCols] = size(strings);
    [numRows, numCols] = size(numbers);
    likelyRow = size(raw,1) - numRows;
    % Break the data up into a new structure with one field per column.
    if strCols == numCols && likelyRow > 0 && strRows >= likelyRow
        newData1.colheaders = strings(likelyRow, :);
    end
end

% Create new variables in the base workspace from those fields.
for i = 1:size(newData1.colheaders, 2)
    assignin('caller', genvarname(newData1.colheaders{i}), newData1.data(:,i));
end
```

```
function importfileDM(fileToRead1)
%IMPORTFILE(FILETOREAD1)
%  Imports data from the specified file
%  FILETOREAD1:  file to read

%  Auto-generated by MATLAB on 30-Jul-2010 14:43:44

% Import the file
newData1 = importdata(fileToRead1);

% For some XLS and other spreadsheet files, returned data are packed
% within an extra layer of structures.  Unpack them.
fields = fieldnames(newData1.data);
newData1.data = newData1.data.(fields{1});
fields = fieldnames(newData1.textdata);
newData1.textdata = newData1.textdata.(fields{1});

% Create new variables in the base workspace from those fields.
vars = fieldnames(newData1);
for i = 1:length(vars)
    assignin('caller', vars{i}, newData1.(vars{i}));
end
```

```
function [parameters_Meth,parameters_195,parameters_BB1,parameters_Dhb,Xf_195,Xf_Meth,Xf_BB1,Xf_Dhb]↙
= inputs

importfileDM('Model_Inputs\BioInputs.xlsx')

parameters_195 = data(:,1);
parameters_Meth = data(1:7,6);
parameters_BB1 = data(1:11,12);
parameters_Dhb = data(1:11,17);
Xf_195 = parameters_195(15);
Xf_Meth = parameters_Meth(7);
Xf_BB1 = parameters_BB1(11);
Xf_Dhb = parameters_Dhb(11);

end
```

```
function PCEmat_analyt = LEanalytical(xL,vx,delx0,Dy_PCE,x_values,y_values,Asat_PCE)
% PCEMAT_ANALYT Calculate the concentration profile of PCE along a pool,
% according to eq. 13 of Seagren et. al  (1994).

% Initialize variables.
n = length(x_values);
m = length(y_values);
PCEmat_analytstar = zeros(m,n);

x_values = x_values - sum(delx0((1:10)));

% Calculate non-dimensional Parameters.  Given that flow is uniform,
% one-dimensional.
vx_A = mean(vx);
Dy_PCE_A = Dy_PCE(1);

Pet = vx_A*xL/Dy_PCE_A;          % Transverse Peclet number
Da2 = 0;                         % No reaction, means Damkohler = 0

xstar = x_values/xL;      % Non-dimensionalized length
ystar = y_values/xL;      % Non-dimensionalized width

for i = 11:n
    for j = 1:m

        PCEmat_analytstar(j,i) = ...
            0.5*exp(-ystar(j)*sqrt(Da2*Pet))*...
            erfc((ystar(j)/(2*sqrt(xstar(i)/Pet))) - sqrt(Da2*xstar(i)))...
            + 0.5*exp(-ystar(j)*sqrt(Da2*Pet))*...
            erfc((ystar(j)/(2*sqrt(xstar(i)/Pet))) + sqrt(Da2*xstar(i)));

    end
end

PCEmat_analyt = PCEmat_analytstar*Asat_PCE;

end
```

```
function ydot = reaction_sys_BB1(y,~,~,parameters_BB1,~)
%REACTION_SYS Set up system of ODE's for the reaction module.
%   Input y as a column vector of concentrations.  The order is
%   [ACE H2 PCE TCE DCE VC ETH X195 XMeth XBB1 XDhb]

%% Assign parameters and y values to useful names.
qmax_BB1_PCE = parameters_BB1(1);   % umol/mgVSS/hr
qmax_BB1_TCE = parameters_BB1(2);   % umol/mgVSS/hr
Ka_BB1_PCE = parameters_BB1(4);     % umol/L
Ka_BB1_TCE = parameters_BB1(5);     % umol/L
Ks_BB1_ACE = parameters_BB1(6);     % umol/L
Y_BB1 = parameters_BB1(7);          % mgVSS/umol_H2
Sd_BB1 = parameters_BB1(8);         % umol_H2
decay_BB1 = parameters_BB1(9);      % hr^(-1)
fe_BB1 = parameters_BB1(10);        % unitless
Xf_BB1 = parameters_BB1(11);

S_ACE = y(1);                       % umol/L
A_PCE = y(3);                       % umol/L
A_TCE = y(4);                       % umol/L
X_BB1 = y(10);                      % mgVSS/L

%% Enter ODE's for H2 and Biomass.
BB1go = S_ACE > Sd_BB1;
PCE_BB1_R = BB1go*qmax_BB1_PCE*X_BB1*...
    (A_PCE/(Ka_BB1_PCE+A_PCE))*((S_ACE-Sd_BB1)/(Ks_BB1_ACE+S_ACE-Sd_BB1));
TCE_BB1_R = BB1go*qmax_BB1_TCE*X_BB1*...
    (A_TCE/(Ka_BB1_TCE+A_TCE))*((S_ACE-Sd_BB1)/(Ks_BB1_ACE+S_ACE-Sd_BB1));

Adot_PCE = -PCE_BB1_R;
Adot_TCE = PCE_BB1_R - TCE_BB1_R;
Adot_DCE = TCE_BB1_R;
Adot_VC = 0;
Adot_ETH = 0;

Sdot_ACE = -0.25*(PCE_BB1_R + TCE_BB1_R)/fe_BB1;
Sdot_H2 = 0;

Xdot_BB1 = (X_BB1 < 0.99*Xf_BB1)*(Y_BB1*(PCE_BB1_R + TCE_BB1_R) - decay_BB1*X_BB1);

Xdot_195 = 0;
Xdot_Meth = 0;
Xdot_Dhb = 0;

%% Enter output into the form necessary for the ODE Solver.
ydot = [
    Sdot_ACE
    Sdot_H2
    Adot_PCE
    Adot_TCE
    Adot_DCE
    Adot_VC
    Adot_ETH
    Xdot_195
    Xdot_Meth
    Xdot_BB1
```

```
    Xdot_Dhb];

end
```

```
function ydot = reaction_sys_BB1_Dhc(y,parameters_195,~,parameters_BB1,~ )
%REACTION_SYS Set up system of ODE's for the reaction module.
%   Input y as a column vector of concentrations.  The order is
%   [ACE H2 PCE TCE DCE VC ETH X195 XMeth XBB1 XDhb]

%% Assign parameters and y values to useful names.
qmax_195_PCE = parameters_195(1);   % umol/mgVSS/hr
qmax_195_TCE = parameters_195(2);   % umol/mgVSS/hr
qmax_195_DCE = parameters_195(3);   % umol/mgVSS/hr
qmax_195_VC = parameters_195(4);    % umol/mgVSS/hr
Ka_195_PCE = parameters_195(6);     % umol/L
Ka_195_TCE = parameters_195(7);     % umol/L
Ka_195_DCE = parameters_195(8);     % umol/L
Ka_195_VC = parameters_195(9);      % umol/L
Ks_195_H2 = parameters_195(10);     % umol/L
Y_195 = parameters_195(11);         % mgVSS/umol_H2
Sd_195 = parameters_195(12);        % umol_H2
decay_195 = parameters_195(13);     % hr^(-1)
fe_195 = parameters_195(14);        % unitless
Xf_195 = parameters_195(15);        % mgVSS/L

qmax_BB1_PCE = parameters_BB1(1);   % umol/mgVSS/hr
qmax_BB1_TCE = parameters_BB1(2);   % umol/mgVSS/hr
Ka_BB1_PCE = parameters_BB1(4);     % umol/L
Ka_BB1_TCE = parameters_BB1(5);     % umol/L
Ks_BB1_ACE = parameters_BB1(6);     % umol/L
Y_BB1 = parameters_BB1(7);          % mgVSS/umol_H2
Sd_BB1 = parameters_BB1(8);         % umol_H2
decay_BB1 = parameters_BB1(9);      % hr^(-1)
fe_BB1 = parameters_BB1(10);        % unitless
Xf_BB1 = parameters_BB1(11);        % mgVSS/L

S_ACE = y(1);                       % umol/L
S_H2 = y(2);                        % umol/L
A_PCE = y(3);                       % umol/L
A_TCE = y(4);                       % umol/L
A_DCE = y(5);                       % umol/L
A_VC = y(6);                        % umol/L
X_195 = y(8);                       % mgVSS/L
X_BB1 = y(10);                      % mgVSS/L

%% Enter ODE's for H2 and Biomass.
Dhcgo = S_H2 > Sd_195;
BB1go = S_ACE > Sd_BB1;
PCEgo = A_PCE > 0;
TCEgo = A_TCE > 0;
DCEgo = A_DCE > 0;
VCgo = A_VC > 0;

PCE_195_R = Dhcgo*PCEgo*qmax_195_PCE*X_195*...
    (A_PCE/(Ka_195_PCE+A_PCE))*((S_H2-Sd_195)/(Ks_195_H2+S_H2-Sd_195));
TCE_195_R = Dhcgo*TCEgo*qmax_195_TCE*X_195*...
    (A_TCE/(Ka_195_TCE+A_TCE))*((S_H2-Sd_195)/(Ks_195_H2+S_H2-Sd_195));
DCE_195_R = Dhcgo*DCEgo*qmax_195_DCE*X_195*...
    (A_DCE/(Ka_195_DCE+A_DCE))*((S_H2-Sd_195)/(Ks_195_H2+S_H2-Sd_195));
VC_195_R = Dhcgo*VCgo*qmax_195_VC*X_195*...
```

```
    (A_VC/(Ka_195_VC+A_VC))*((S_H2-Sd_195)/(Ks_195_H2+S_H2-Sd_195));

PCE_BB1_R = BB1go*PCEgo*qmax_BB1_PCE*X_BB1*...
    (A_PCE/(Ka_BB1_PCE+A_PCE))*((S_ACE-Sd_BB1)/(Ks_BB1_ACE+S_ACE-Sd_BB1));
TCE_BB1_R = BB1go*TCEgo*qmax_BB1_TCE*X_BB1*...
    (A_TCE/(Ka_BB1_TCE+A_TCE))*((S_ACE-Sd_BB1)/(Ks_BB1_ACE+S_ACE-Sd_BB1));

Adot_PCE = -PCE_195_R - PCE_BB1_R;
Adot_TCE = PCE_195_R - TCE_195_R + PCE_BB1_R - TCE_BB1_R;
Adot_DCE = TCE_195_R - DCE_195_R + TCE_BB1_R;
Adot_VC = DCE_195_R - VC_195_R;
Adot_ETH = VC_195_R;

Sdot_ACE = -0.25*(PCE_BB1_R + TCE_BB1_R)/fe_BB1;
Sdot_H2 = -(PCE_195_R + TCE_195_R + DCE_195_R + VC_195_R)/fe_195;

if (X_195/Xf_195 + X_BB1/Xf_BB1)<=0.99
    Xdot_195 = Y_195*(PCE_195_R + TCE_195_R + DCE_195_R) - decay_195*X_195;
    Xdot_BB1 = Y_BB1*(PCE_BB1_R + TCE_BB1_R) - decay_BB1*X_BB1;
else
    omega = min(abs((decay_195*X_195 + decay_BB1*X_BB1)/...
        (Y_195*(PCE_195_R + TCE_195_R + DCE_195_R) + Y_BB1*(PCE_BB1_R + TCE_BB1_R))),1);

    Xdot_195 = omega*Y_195*(PCE_195_R + TCE_195_R + DCE_195_R) - decay_195*X_195;
    Xdot_BB1 = omega*Y_BB1*(PCE_BB1_R + TCE_BB1_R) - decay_BB1*X_BB1;
end

Xdot_Meth = 0;
Xdot_Dhb = 0;

%% Enter output into the form necessary for the ODE Solver.
ydot = [
    Sdot_ACE
    Sdot_H2
    Adot_PCE
    Adot_TCE
    Adot_DCE
    Adot_VC
    Adot_ETH
    Xdot_195
    Xdot_Meth
    Xdot_BB1
    Xdot_Dhb];

if sum(isnan(ydot)) > 0
    disp('Break in the BioRxn')
end

end
```

```
function ydot = reaction_sys_Dhc(y,parameters_195,~,~,~ )
%REACTION_SYS Set up system of ODE's for the reaction module.
%   Input y as a column vector of concentrations.  The order is
%   [ACE H2 PCE TCE DCE VC ETH X195 XMeth]

%% Assign parameters and y values to useful names.
qmax_195_PCE = parameters_195(1);   % umol/mgVSS/hr
qmax_195_TCE = parameters_195(2);   % umol/mgVSS/hr
qmax_195_DCE = parameters_195(3);   % umol/mgVSS/hr
qmax_195_VC = parameters_195(4);    % umol/mgVSS/hr
Ka_195_PCE = parameters_195(6);     % umol/L
Ka_195_TCE = parameters_195(7);     % umol/L
Ka_195_DCE = parameters_195(8);     % umol/L
Ka_195_VC = parameters_195(9);      % umol/L
Ks_195_H2 = parameters_195(10);      % umol/L
Y_195 = parameters_195(11);         % mgVSS/umol_H2
Sd_195 = parameters_195(12);        % umol_H2
decay = parameters_195(13);         % hr^(-1)
fe = parameters_195(14);            % unitless
Xf_195 = parameters_195(15);        % mgVSS/L

S_H2 = y(2);                        % umol/L
A_PCE = y(3);                       % umol/L
A_TCE = y(4);                       % umol/L
A_DCE = y(5);                       % umol/L
A_VC = y(6);                        % umol/L
X_195 = y(8);                       % mgVSS/L

%% Enter ODE's for H2 and Biomass.
PCE_195_R = qmax_195_PCE*X_195*(S_H2>=Sd_195)*...
    (A_PCE/(Ka_195_PCE+A_PCE))*((S_H2-Sd_195)/(Ks_195_H2+S_H2-Sd_195));
TCE_195_R = qmax_195_TCE*X_195*(S_H2>=Sd_195)*...
    (A_TCE/(Ka_195_TCE+A_TCE))*((S_H2-Sd_195)/(Ks_195_H2+S_H2-Sd_195));
DCE_195_R = qmax_195_DCE*X_195*(S_H2>=Sd_195)*...
    (A_DCE/(Ka_195_DCE+A_DCE))*((S_H2-Sd_195)/(Ks_195_H2+S_H2-Sd_195));
VC_195_R = qmax_195_VC*X_195*(S_H2>=Sd_195)*...
    (A_VC/(Ka_195_VC+A_VC))*((S_H2-Sd_195)/(Ks_195_H2+S_H2-Sd_195));

Adot_PCE = -PCE_195_R;
Adot_TCE = PCE_195_R - TCE_195_R;
Adot_DCE = TCE_195_R - DCE_195_R;
Adot_VC = DCE_195_R - VC_195_R;
Adot_ETH = VC_195_R;

Sdot_ACE = 0;
Sdot_H2 = -(PCE_195_R + TCE_195_R + DCE_195_R + VC_195_R)/fe;

Xdot_195 = (X_195 < 0.99*Xf_195)*(Y_195*(PCE_195_R + TCE_195_R + DCE_195_R) - decay*X_195);

Xdot_Meth = 0;
Xdot_BB1 = 0;
Xdot_Dhb = 0;

%% Enter output into the form necessary for the ODE Solver.
ydot = [
    Sdot_ACE
```

```
    Sdot_H2
    Adot_PCE
    Adot_TCE
    Adot_DCE
    Adot_VC
    Adot_ETH
    Xdot_195
    Xdot_Meth
    Xdot_BB1
    Xdot_Dhb];
end
```

Appendix B: Model Input Spreadsheets

Table B.1: ScenarioInputs.xlsx

Physical Parameters				Influent Concentrations							
Length (Flow direction)	5	cm		Acetate	5000	µmol/L					
Width (transverse direction)	5	cm		H2	0	µmol/L					
thickness (no movement)	0.0035	cm		PCE	0	µmol/L					
porosity (n)	0.39			TCE	0	µmol/L					
Volumetric flow rate (Q)	0.016	mL/hr		DCE	0	µmol/L					
Longitudinal dispersivity (αL)	0.03	cm		VC	0	µmol/L					
Transverse dispersivity (αT)	0.000803	cm		Ethene	0	µmol/L		Simulation duration	672	hr	
Tortuosity (τ)	0.12685							OS Time Step	0.00390625	hr	
Initial Hydraulic Conductivity (K0)	1	cm/hr		Chemical Diffusivity (25 C)				Max AD Time Step	9.76563E-05	hr	
Min. Hydraulic	0.005	cm/h		Acetate	1.29E-	m2/sec	m2/sec	Report Interval	6	hr	g

Conductivity (Kmin)		r			09		chosen to work with StEPP				o o d
Hydraulic potential at outlet (hL)	1	cm		H2	2.17E-09	m2/sec					
Biofilm (1) or plugs (2)	1			PCE	8.87E-10	m2/sec		Initial Biomass Conc.	0.15	mgV SS/L	
Local Equilibrium (1) or NLE (2)	1			TCE	9.74E-10	m2/sec		Abiotic (0), Dhc (1), Meth (2), BB1 (3), Dhc+Meth (4), Dhc+BB1 (5), Meth+BB1 (6), Dhc+BB1+Meth (7)	3		
Mass Transfer Coefficient (k)	2.916666667	cm/hr		DCE	1.09E-09	m2/sec					
				VC	1.25E-09	m2/sec					
				Ethene	9.02E-10	m2/sec					
				PCE Sat.	904.5	µmol/L					
				kl	0.7	m/d					

Table B.2: DiscretizationInputs.xlsx

xL	deltaX	cumulative		yL	deltaY	cumulative
5	0.01	0.01		5	0.0025	0.0025
	0.01	0.02			0.0025	0.005
	0.01	0.03			0.0025	0.0075
	0.01	0.04			0.0025	0.01
	0.01	0.05			0.0025	0.0125
	0.01	0.06			0.0025	0.015
	0.01	0.07			0.0025	0.0175
	0.01	0.08			0.0025	0.02
	0.01	0.09			0.0025	0.0225
	0.01	0.1			0.0025	0.025
	0.01	0.11			0.0025	0.0275
	0.0105	0.1205			0.0025	0.03
	0.011025	0.131525			0.0025	0.0325
	0.01157625	0.14310125			0.0025	0.035
	0.012155063	0.155256313			0.0025	0.0375
	0.012762816	0.168019128			0.0025	0.04
	0.013400956	0.181420085			0.0025	0.0425
	0.014071004	0.195491089			0.0025	0.045
	0.014774554	0.210265643			0.0025	0.0475
	0.015513282	0.225778925			0.0025	0.05
	0.016288946	0.242067872			0.002625	0.052625
	0.017103394	0.259171265			0.00275625	0.05538125
	0.017958563	0.277129828			0.002894063	0.058275313
	0.018856491	0.29598632			0.003038766	0.061314078

	0.019799316	0.315785636			0.003190704	0.064504782
	0.020789282	0.336574918			0.003350239	0.067855021
	0.021828746	0.358403664			0.003517751	0.071372772
	0.022920183	0.381323847			0.003693639	0.075066411
	0.024066192	0.405390039			0.003878321	0.078944731
	0.025269502	0.430659541			0.004072237	0.083016968
	0.026532977	0.457192518			0.004275848	0.087292816
	0.027859626	0.485052144			0.004489641	0.091782457
	0.029252607	0.514304751			0.004714123	0.09649658
	0.030715238	0.545019989			0.005185535	0.101682115
	0.032250999	0.577270988			0.005704089	0.107386204
	0.033863549	0.611134538			0.006274498	0.113660701
	0.035556727	0.646691264			0.006901947	0.120562649
	0.037334563	0.684025828			0.007592142	0.128154791
	0.039201291	0.723227119			0.008351356	0.136506147
	0.041161356	0.764388475			0.009186492	0.145692639
	0.043219424	0.807607899			0.010105141	0.15579778
	0.045380395	0.852988294			0.011115655	0.166913435
	0.047649415	0.900637708			0.012227221	0.179140655
	0.05	0.950637708			0.013449943	0.192590598
	0.05	1.000637708			0.014794937	0.207385535
	0.05	1.050637708			0.016274431	0.223659966
	0.05	1.100637708			0.017901874	0.241561839
	0.05	1.150637708			0.019692061	0.2612539
	0.05	1.200637708			0.021661267	0.282915168
	0.05	1.250637708			0.023827394	0.306742561
	0.05	1.300637708			0.026210133	0.332952695

	0.05	1.350637708			0.028831147	0.361783841
	0.05	1.400637708			0.031714261	0.393498103
	0.05	1.450637708			0.034885687	0.42838379
	0.05	1.500637708			0.038374256	0.466758046
	0.05	1.550637708			0.042211682	0.508969728
	0.05	1.600637708			0.04643285	0.555402578
	0.05	1.650637708			0.051076135	0.606478713
	0.05	1.700637708			0.056183748	0.662662461
	0.05	1.750637708			0.061802123	0.724464584
	0.05	1.800637708			0.067982336	0.79244692
	0.05	1.850637708			0.074780569	0.867227489
	0.05	1.900637708			0.075	0.942227489
	0.05	1.950637708			0.075	1.017227489
	0.05	2.000637708			0.075	1.092227489
	0.05	2.050637708			0.075	1.167227489
	0.05	2.100637708			0.075	1.242227489
	0.05	2.150637708			0.075	1.317227489
	0.05	2.200637708			0.075	1.392227489
	0.05	2.250637708			0.075	1.467227489
	0.05	2.300637708			0.075	1.542227489
	0.05	2.350637708			0.075	1.617227489
	0.05	2.400637708			0.075	1.692227489
	0.05	2.450637708			0.075	1.767227489
	0.05	2.500637708			0.075	1.842227489
	0.05	2.550637708			0.075	1.917227489
	0.05	2.600637708			0.075	1.992227489
	0.05	2.650637708			0.075	2.067227489

	0.05	2.700637708			0.075	2.142227489
	0.05	2.750637708			0.075	2.217227489
	0.05	2.800637708			0.075	2.292227489
	0.05	2.850637708			0.075	2.367227489
	0.05	2.900637708			0.075	2.442227489
	0.05	2.950637708			0.075	2.517227489
	0.05	3.000637708			0.075	2.592227489
	0.05	3.050637708			0.075	2.667227489
	0.05	3.100637708			0.075	2.742227489
	0.05	3.150637708			0.075	2.817227489
	0.05	3.200637708			0.075	2.892227489
	0.05	3.250637708			0.075	2.967227489
	0.05	3.300637708			0.075	3.042227489
	0.05	3.350637708			0.075	3.117227489
	0.05	3.400637708			0.075	3.192227489
	0.05	3.450637708			0.075	3.267227489
	0.05	3.500637708			0.075	3.342227489
	0.05	3.550637708			0.075	3.417227489
	0.05	3.600637708			0.075	3.492227489
	0.05	3.650637708			0.075	3.567227489
	0.05	3.700637708			0.075	3.642227489
	0.05	3.750637708			0.075	3.717227489
	0.05	3.800637708			0.075	3.792227489
	0.05	3.850637708			0.075	3.867227489
	0.05	3.900637708			0.075	3.942227489
	0.05	3.950637708			0.075	4.017227489
	0.05	4.000637708			0.075	4.092227489

	0.05	4.050637708			0.075	4.167227489
	0.05	4.100637708			0.075	4.242227489
	0.05	4.150637708			0.075	4.317227489
	0.05	4.200637708			0.075	4.392227489
	0.05	4.250637708			0.075	4.467227489
	0.05	4.300637708			0.075	4.542227489
	0.05	4.350637708			0.075	4.617227489
	0.05	4.400637708			0.075	4.692227489
	0.05	4.450637708			0.075	4.767227489
	0.05	4.500637708			0.075	4.842227489
	0.05	4.550637708			0.075	4.917227489
	0.05	4.600637708			0.082772511	5
	0.05	4.650637708				
	0.05	4.700637708				
	0.05	4.750637708				
	0.05	4.800637708				
	0.05	4.850637708				
	0.05	4.900637708				
	0.05	4.950637708				
	0.049362292	5				

Table B.3: BioInputs.xlsx

Dhc. mc-carty 195					Hydrogeno-trophic Methano-gen				*Dsm. michiganen-sis* (BB1)				*Dhb. restrict-us*				
q_max	PCE	6.8	µmol/mg VSS/hr	H2	q_max	40	µmol/mgVSS/hr	Fennell et al. (1998)	q_max	PCE	12.6	µmol/mg VSS/hr	q_max	PCE	22.5	µmol/mg VSS/hr	
	TCE	7.9	µmol/mg VSS/hr		Ks	0.5	µmol/L			TCE	17	µmol/mg VSS/hr		TCE	29.8	µmol/mg VSS/hr	
	DCE	13.2	µmol/mg VSS/hr		Y	0.00143	mgVSS/µmol H2			Acetate	4.66	µmol/mg VSS/hr		H2	56.4	µmol/mg VSS/hr	
	VC	0.9	µmol/mg VSS/hr		Sd H2	0.008	µmol/L		KS	PCE	9.31	µmol/L	KS	PCE	7.2	µmol/L	
	H2	12.8	µmol/mg VSS/hr		decay	0.001	hr-1			TCE	2.83	µmol/L		TCE	1.3	µmol/L	
KS	PCE	21.5	µmol/L		fe	0.8877				Acetate	5.77	µmol/L		H2	3.3	µmol/L	
	TCE	29	µmol/L		Xf	8000	mg VSS/L			Yield	0.0033	mg VSS/µmol Cl-		Yield	0.0044	mg VSS/µmol Cl-	
	DCE	33.6	µmol/L							Sd Acet.	0.41	µmol/L		Sd H2	0.00187	µmol/L	
	VC	637	µmol/L							decay	0.0054	hr-1		decay	0.004	hr-1	same as Dhc

	H2	0.03	µmol/L							fe	0.95			fe	0.8		(abritrarily chosen)
	Yield	0.0047	mg VSS/µmol Cl-							Xf	8000	mg VSS/L		Xf	8000	mg VSS/L	
	Sd H2	0.00187	µmol/L														
	decay	0.004	hr-1														
	fe	0.738															
	Xf	8000	mg VSS/L														

Appendix C: Code Modification FAQs

1. *How do I change the number of CPUs the simulation will use?*

   In line 6 of *simulation_master.m* set the number after "matlabpool" equal to the number of CPUs to be used (max 8 by Matlab, otherwise max is the number of CPUs in the given machine).
2. *How do I add another chemical species to be tracked?*

   First, the new chemical needs to be entered into the *ScenarioInputs.xlsx* spreadsheet. It will need to have both an inlet concentration (middle column, top half) and a chemical diffusivity (middle column, bottom half). For the first new chemical added, the other values do not need to be moved. However, any moving of other values will need to be taken care of in the Matlab code files.
   Next, some changes will need to be made to *initializesimulation.m*.
   - Line 6: Change the value of Nchem to reflect the new number of chemical species.
   - Line 29: Ensure that PCE remains third in the order of calculations in all calculations, or change the input to whichever location PCE will be.
   - Lines 31-45: Add a line for each new chemical species influent concentration and chemical diffusivity. Also, verify that the data being read from the other lines is lining up with the correct locations if any inputs on the spreadsheet were moved. For reference, the matrix *data* has its top-left element at location B2 in the spreadsheet.
   - Lines 100-106: Create the initialization vector following the pattern already established.
   - Lines 210-216: Initialize the value of the concentration vector following the pattern already established.
   - Lines 257-343: Initialize and calculate Dx and Dy vectors following the pattern already established.

- Lines 347-371: Initialize efflux tracking vectors and matrices following the pattern already established.

There are also some changes that must be made to *simulation_master.m*.

- Line 11: Change Nchem to the correct number of chemical species.
- Lines 42-63: Add a line to each block for the new chemical following the established pattern. Also add the new chemical to efflux1 and efflux2
- Lines 93-100: Add a line for the new chemical species following the pattern already established.
- Lines 110-137: Add a line for the new chemical species in each block following the pattern already established. Maintain consistent order throughout.
- Lines 218-225, 372-379: Add a line for the new chemical species following the pattern already established.
- Lines 383-399: Add a line for the new chemical species in each block following the pattern already established.
- Lines 418-424: Add a line for the new chemical species following the pattern already established.
- Lines 490-562: Add a line for the new chemical species in each block following the pattern already established.
- Lines 584-605: Add a line for the new chemical species in each block following the pattern already established.

3. *How do I add a new microbial species?*

It is a complicated process, so begin with seeing if the new microbial species can instead be simulated by simply changing the microbial kinetics of the species already simulated.

4. *How do I identify numerical instability, and what can be done?*

5. *How do I start a simulation mid-run from a saved file?*

    - First, find the .mat in the *Workspace* folder under *Results*.
    - Next, copy the .mat file into the *Model_Inputs* folder.
    - Rename the file to *Initialization.mat.*
    - Run *simulation_master* in the Matlab command window. It will automatically resume from the saved location.
    - Note: The results will be saved in a new folder, so keep good documentation to combine separate pieces of the same simulation together.