RAFT

A Simulator for **ReA**ctive
Flow and **T**ransport of Groundwater Contaminants

A. Chilakapati

July 1995

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RL0 1830

Pacific Northwest Laboratory
Richland, Washington 99352

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

COPYRIGHT AND TRADEMARK ACKNOWLEDGEMENTS

RAFT : Copyright ©1993 by A. Chilakapati.

RAFT : Developed by Battelle Memorial Institute, Pacific Northwest Laboratory.

DEC ALPHA is a registered trademark of Digital Equipment Corporation.

IBM is registered trademark of International Business Machines Corporation.

MAPLE is a registered trademark of the 766884 Ontario Inc.

Sun is a registered trademark of Sun Microsystems, Inc.

UNIX is a registered trademark of AT&T Information Systems.

DISCLAIMER

**Portions of this document may be illegible
in electronic image products. Images are
produced from the best available original
document.**

Errata

Since this report was written, RAFT has undergone a lot of changes so that this report does not any longer reflect the current status of RAFT. An updated report is expected to be placed on WWW in the near future. Please send an e-mail to

a_chilakapati@pnl.gov

if you want to be notified when the report becomes available on WWW.

MASTER

df

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Summary

This report documents the use of the simulator RAFT for the ReActive Flow and Transport of groundwater contaminants. RAFT can be used as a predictive tool in the design and analysis of laboratory and field experiments or it can be used for the estimation of model/process parameters from experiments. RAFT simulates the reactive transport of groundwater contaminants in one-, two-, or three-dimensions and it can model user specified source/sink configurations and arbitrary injection strategies. A suite of solvers for transport, reactions and regression are employed so that a combination of numerical methods best suited for a problem can be chosen. User specified coupled equilibrium and kinetic reaction systems can be incorporated into RAFT. RAFT is integrated with a symbolic computational language MAPLE, to automate code generation for arbitrary reaction systems.

RAFT is expected to be used as a simulator for engineering design for field experiments in groundwater remediation including bioremediation, reactive barriers and redox manipulation. As an integrated tool with both the predictive ability and the ability to analyze experimental data, RAFT can help in the development of remediation technologies, from laboratory to field.

KEYWORDS: contaminant transport, groundwater modeling, bioremediation, saturated transport, reactive flow, numerical modeling, differential-algebraic systems.

Acknowledgements

A part of this work is funded by the Office of Technology Development, within the Department of Energy's Office of Environmental Management, under "Plumes Focus Area - Organics Product Line" (formerly known as the VOCs in Arid Soils Integrated Demonstration).

A part of this work was prepared with the support of the following contributors:

Headquarters: Office of Technology Development
 Plumes Focus Area - Organics Product Line
 Rashalee Levine, Program Manager
 (formerly David L. Biancosino)

Operations Office: Richland Operations Office
 Technology Development Division
 Roger Christensen, Technical Program Officer
 (formerly Deborah E. Trader)

Contractor: Pacific Northwest Laboratory
 Environmental Management Directorate
 Steven C. Slate, Technical Program Manager

A part of this work was also funded by the Subsurface Science Program, Office of Health and Environmental Research, U.S. Department of Energy.

Also the Environmental Dynamics and Simulation Task of the Environmental Molecular Sciences Laboratory (EMSL) Construction Project at the Pacific Northwest Laboratory contributed in part to the development of this simulator.

Nomenclature

Symbol	Description - Units
c_i	Concentration of the i th component. M/L^3 .
c_i^o	Concentration of the i th component just outside an inlet boundary.
c_i^I	Initial concentration of the i th component. M/L^3 .
$\tilde{c}_i$	The injection concentration of the i th component at an injection well.
$\underline{\underline{D}}$	Dispersion tensor. L^2/T .
$\underline{\underline{I}}$	Identity matrix.
L_x, L_y, L_z	Lengths in the X, Y and Z directions. L .
ψ	Potential. L .
R_i	Retardation factor for the i th component.
$\vec{V}$	Velocity vector. L/T .
$ \vec{V} $	Magnitude of the velocity vector. L/T .
V_x, V_y, V_z	The velocities in the x, y , and z directions. L/T .
$\{c_n\}$	The set of concentrations for all the components.
d_m	Molecular diffusivity. L^2/T .
$\underline{\underline{k}}^h$	Hydraulic conductivity tensor. L/T .
k_x^h, k_y^h, k_z^h	Hydraulic conductivity in the x, y, z directions. L/T .
q	Source/Sink strength. $1/T$.
Q_w	The total flow-rate in/out of a well. L^3/T .

$\hat{q}_{wk}$	Flow-rate in/out of a well-block. L^3/T .
$\underline{r}$	Position vector of a point. L .
r_i	Rate of production of the i th component. $M/(L^3T)$.
t_1, t_2	Points on the time-axis. $t_2 - t_1 = \Delta t$. T .
ϕ	Porosity.
α_T	Transverse dispersivity. L .
α_L	Longitudinal dispersivity. L .
$\vec{\nu}$	Unit outward normal to the boundary.
$\Delta x_i, dx, \Delta x$	The grid-size in the x -direction. $x_{i+1} - x_i$. L .
$\Delta y_j, dy, \Delta y$	The grid-size in the y -direction. $y_{j+1} - y_j$. L .
$\Delta z_k, dz, \Delta z$	The grid-size in the z -direction. $z_{k+1} - z_k$. L .
Δt	The size of the time-step. T .
CMM	Characteristic-Mixed Method.
CCM	Characteristic-Conservative Method.
FCT	Flux-Corrected Transport method.
TVD	Total Variation Diminishing method.

Contents

Summary	iii
Acknowledgements	v
Nomenclature	vii
1 Introduction	1
2 Equations	3
3 RAFT	6
3.1 Input Data	7
3.1.1 Description of the file <code>data</code>	9
3.1.2 Description of the file <code>columndata</code>	20
3.1.3 Description of the file <code>welldata</code>	22
3.1.4 Description of the file <code>sizes.h</code>	26
3.2 Input Files	28
3.3 Output Files	30
3.4 Reactions	34
3.4.1 Formulation of the reaction system	36
3.4.2 Solution of initial-value DAE systems	44
3.4.3 Preparation of the file <code>rxns.f</code>	47
3.4.4 Preparation of the file <code>preeqrns.f</code>	49
3.4.5 Description of the file <code>datarxn</code>	50
3.5 Parameter Estimation	62
3.6 Library Packages	66
3.7 Summary	67
4 Examples	74
4.1 Tracer Transport through a Column	74
4.1.1 Constant injection	74
4.1.2 Pulsing square wave. Pure advection	81
4.2 Reactive Transport in One-Dimension	84

4.3	Reactive Transport in Two-Dimensions	89
4.4	Sources and Sinks	98
4.4.1	Single injection well	98
4.4.2	A recirculation well	104
4.5	Coupled Reaction Systems	108
A	Specification of Reactions	118
	Bibliography	122

List of Figures

3.1	The tensor grid	7
3.2	Locations for velocities	35
3.3	Choosing a Solver for Reactions	72
4.1	Linear displacement	79
4.2	Numerical/Analytic Solutions	79
4.3	Convergence with the time-step size	80
4.4	Convergence with the grid size	81
4.5	Convergence with the grid size	82
4.6	Numerical/Analytic Solutions	83
4.7	Numerical/Analytic Solutions	84
4.8	Breakthrough of a pulsing square wave	85
4.9	Breakthrough of a pulsing square wave	85
4.10	Breakthrough of a pulsing square wave	86
4.11	Breakthrough of a pulsing square wave	86
4.12	Breakthrough of a pulsing square wave	87
4.13	Transport of a reactive tracer	89
4.14	Breakthrough of a reactive tracer	90
4.15	Breakthrough of a reactive tracer	90
4.16	Breakthrough of a reactive tracer	91
4.17	Conductivity	97
4.18	Microbial degradation	98
4.19	Single injection well	102
4.20	Pulse injection in a well	104
4.21	Pulse injection in a well	105
4.22	Pulse injection in a well	105
4.23	A dual screened recirculation well	106
4.24	Breakthrough Curves. Dual screened well	109
4.25	A mixed reaction system	111
4.26	A mixed reaction system	112
4.27	A mixed reaction system with transport	114
4.28	A mixed reaction system with transport	115
4.29	A DAE system with transport	117

Chapter 1

Introduction

RAFT (ReActive Flow and Transport) is a computer code that solves the equations that describe (in three dimensions) the reactive flow and transport of solutes in the saturated zone. The flow is taken to be incompressible and unaffected by the variations in concentrations of solutes in groundwater. Velocities are given by Darcy's law, and the concentrations are taken to obey the advection-diffusion-reaction equation. Reactions can have a kinetic or equilibrium description. RAFT provides three different methods for the solution of the advection-diffusion-reaction equation. In the first method, a characteristic-conservative method (CCM) is used to obtain the velocity field and a characteristic-mixed method (CMM) is used for concentrations. In the second method, a cell-centered finite difference scheme is used to obtain velocities and a flux-corrected transport (FCT) method is used for concentrations. In the third method, a cell-centered finite difference scheme is used to obtain velocities and a third-order Leonard method with a total variation diminishing (TVD) property is used for concentrations. The **second and third methods (FCT & TVD)** are the **recommended** methods for most problems since they are at least an order of magnitude faster compared to CMM (especially in multidimensions, ≥ 2). FCT and TVD schemes require the Courant number to be less than 0.5 (or even smaller for some complex flow problems). Usually this is not a constraint since the reactions may anyway require small time-step sizes (especially when they are fast or in equilibrium). When the problem allows the use of large time-step sizes (Courant number > 1.0), the first method (CCM/CMM) may be advantageous. In all three methods the reactions are treated with operator-splitting. Mixed kinetic/equilibrium systems can be solved by the reactions module. RAFT can also be used as a tool in parameter estimation. Estimation of parameters is done by a nonlinear regression to experimental observations. The parameters estimated are usually the dispersivities, retardation factors and reaction constants. With some modifications to the code, this facility for optimization can also be used in selecting an optimal design for a groundwater remediation system. For example the optimal well locations, pumping strategy and injection strategy for nutrients can be selected for the bioremediation of contaminated groundwater.

As a research tool RAFT has been used in the evaluation of several numerical meth-

ods for solving contaminant transport problems. Numerical schemes for transport are typically tested for numerical dispersion, wiggles, accuracy and computational efficiency. Schemes for the solution of differential-algebraic systems, for the solution of systems of nonlinear equations and for nonlinear regression are tested for robustness, accuracy and speed. RAFT has been extensively used in the design and analysis of laboratory experiments. RAFT is used in the the development of reaction models and the estimation of reaction parameters from batch and column experiments. RAFT is used in the design of intermediate scale experiments to study the effect of physical and chemical heterogeneities on reactive transport. RAFT is also being used in the design of field experiments to study the migration of microbes in the subsurface. A part of RAFT was developed for use in the design of a field bioremediation experiment at Hanford, under the VOC-Arid Integrated Demonstration program.

This document describes the structure and the use of RAFT. It is assumed that the user is familiar with the subject, has some understanding of numerical methods and has some experience with FORTRAN. A summary of subsurface hydrological and hydrochemical models is given by Mangold and Tsang [26]. An approach to formulate the mixed kinetic/equilibrium reactions with transport is presented by Friedly and Rubin [13, 12]. Yeh and Tripathi [35] present a review of the hydrogeochemical transport models and solution techniques. Recently Yeh and Iskra [34] developed KEMOD for the solution of mixed kinetic and equilibrium models under batch conditions.

Chapter 2 lists the governing equations and briefly describes the three numerical methods employed by RAFT. The user is referred to literature for detailed descriptions of CCM/CMM, FCT, and TVD. Chapter 3 describes the structure of the code and the input/output files. A helpful summary of the use of RAFT appears at the end of this chapter. In Chapter 4 a few sample problems that illustrate the use of RAFT are solved. A few problems with analytic solutions are solved to demonstrate the convergence of the numerical solution to the analytic solution.

Conventions To help locate and identify information in this guide, the following visual cues are used. These are applicable to chapters 3 and 4.

Type style	Used for
infilename	Input filenames
<u>outfilename</u>	Output filenames
<i>invarname</i>	Variable names in the input files
<i>outvarname</i>	Variable names in the output files
<code>data, results, code</code>	The contents of input files, output files and segments of code

Chapter 2

Equations

Transport of contaminants in groundwater is most often described by the advection-diffusion-reaction equation. The flow is usually taken to be incompressible and unaffected by the variations in concentrations of the solutes in groundwater. If c_i is the concentration of the i th component $[M/L^3]$, q is the fluid source/sink strength (positive for a sink and negative for a source, $1/T$) $\vec{V}$ is the darcy velocity field $[L/T]$, ψ is the potential $[L]$, ϕ is the porosity, R_i is the retardation factor for the i th component, $\underline{k}^h$ is the hydraulic conductivity $[L/T]$, $r_i(\{c_n\})$ is the rate of production by reaction for the i th component $[M/TL^3]$ and $\underline{D}$ is the dispersion tensor $[L^2/T]$ then the equations describing the problem can be written as [5],

$$\vec{V} = -\underline{k}^h \vec{\nabla} \psi, \quad (2.1)$$

$$\vec{\nabla} \cdot \vec{V} = -q, \quad (2.2)$$

$$R_i \phi \frac{\partial c_i}{\partial t} + \vec{\nabla} \cdot (c_i \vec{V}) = \vec{\nabla} \cdot [\underline{D} \vec{\nabla} c_i] + R_i \phi r_i(\{c_n\}) - q \tilde{c}_i, \quad (2.3)$$

$$\underline{D} = (\alpha_T |\vec{V}| + \phi d_m) \underline{I} + \frac{(\alpha_L - \alpha_T) \vec{V} \vec{V}}{|\vec{V}|}. \quad (2.4)$$

$\tilde{c}_i$ is the injection concentration for the source and for the sink, it is equal to c_i . α_L is the longitudinal dispersivity, α_T is the transverse dispersivity and d_m is the molecular diffusivity. Here we take $\underline{k}^h$ to be at most a diagonal tensor if not isotropic. The domain of the problem is restricted here to be rectangular. Constant potential or noflow boundary conditions are typical for the velocity problem. For the concentration problem, Neumann (finite dispersive flux) condition is used at outflow boundaries, and a Danckwerts (third type, continuity of total flux, dispersive + convective) boundary condition is used at inflow boundaries.

$$\vec{V} \cdot \nu = 0, \text{ noflow boundary}, \quad (2.5a)$$

$$\psi = \psi_0, \text{ constant potential boundary}, \quad (2.5b)$$

$$[\underline{D} \vec{\nabla} c_i] \cdot \nu = f(x, y, z, t), \text{ finite dispersive flux at outflow boundary}, \quad (2.5c)$$

$$(\vec{V} c_i - [\underline{D} \vec{\nabla} c_i]) \cdot \nu = \vec{V} c_i^o \cdot \nu, \text{ inlet boundary}. \quad (2.5d)$$

Here ν is the unit outward normal to the boundary, c_i^o is the concentration just outside of an inlet boundary. Any concentration profile c_i^I can be used as the initial condition. When some of the species are in equilibrium then the initial condition should reflect it.

Numerical Methods Three different numerical methods are coded into RAFT for the solution of the system of equations 2.1 through 2.5d. The first one is based on the Method of Characteristics (CCM/CMM). The second method implements a version of Flux-Corrected Transport (FCT). The third method implements the third-order Leonard scheme with a total variation diminishing property (TVD). All three methods treat the reactions with operator-splitting.

In the first method [2, 1, 9] the solution procedure for velocities is coupled to the procedure for concentrations, resulting in a direct numerical approximation to the transport theorem. The incompressible darcy velocity field is obtained by a characteristic-conservative method that explicitly conserves the volume of fluid elements. The advection dominated advection-diffusion-reaction equation is solved with the characteristic-mixed method. Thus the fluid elements are followed in time and space for each component along their volume preserving streamlines and the dispersive, reactive contributions to the change in concentrations are evaluated. The combined scheme is locally and globally conservative. It has been implemented in three dimensions and convergence is established.

In the second method, a cell-centered finite difference scheme is used to obtain velocities and a flux-corrected transport method [7, 23, 36, 17] is used for concentrations. In FCT, a low order non-oscillatory solution is initially computed. This solution is usually excessively diffusive. Anti-diffusive fluxes are then computed from higher-order estimates for fluxes. These anti-diffusive fluxes are then "limited" so that they do not create artificial minima/maxima when added to the diffused solution. The final solution is computed by adding these limited amounts of anti-diffusive fluxes to the initially diffused solution.

In the third method, a cell-centered finite difference scheme is used to obtain velocities and a third-order Leonard scheme with a total variation diminishing property [14, 33, 31, 25] is used for concentrations. The following is an excerpt from [14].

For convection-dominated flow, sharp resolution of discontinuities has been achieved by adding a limited anti-diffusive flux to an entropy-satisfying scheme. A new flux limiter is introduced that remains third-order accurate in space and time in smooth regions and results in oscillation-free numerical solutions by imposing the total variation diminishing (TVD) criteria.

Reactions are treated with operator-splitting in all the three methods. The reaction

module can solve coupled systems of equilibrium and kinetic reactions. A suite of solvers for the initial value ordinary differential equation systems (ODES) (when all the reactions have a kinetic description) and for the initial value differential-algebraic equation systems (DAES) (when some reactions have a kinetic description and some others are in equilibrium) are coded into RAFT [19, 20]. The following table lists these solvers and their use in RAFT.

RK4	This subroutine is for the solution of non-stiff ODES. It employs the fourth order Runge-Kutta method.
LSODA	This program is for the solution of ODES. This is a part of ODEPACK. It can automatically detect stiffness and thus employ a suitable numerical method for integration [18].
RADAU5	This program is for the solution of ODES or DAES. It employs implicit Runge-Kutta techniques [15].
DASSL	This program is for the solution of ODES or DAES. It employs backward differentiation formulae [29, 8].
LSODI	This program is for the solution of ODES or DAES [18]. This is a part of ODEPACK.
INTBIS	This program is for the solution of a system of nonlinear equations. This is used by RAFT in the speciation part. INTBIS employs a generalized interval bisection method [22].
NNES	This program is for the solution of a system of nonlinear equations. This is used by RAFT in the speciation part. NNES employs a variety of techniques based on newton methods and minimization [4].

Parameter estimation is done by nonlinear regression to experimental observations. Dispersivities, retardation factors and the reaction constants are usually the parameters estimated by RAFT. RAFT uses the following programs for that purpose.

DQED	This is a program for nonlinear regression by least squares [16].
NL2SOL	This is a program for nonlinear regression by least squares [10].
LMDIF	This is a program from MINPACK [27].

The following linear algebra subroutines are used by many of the above programs.

BLAS	Basic Linear Algebra Subroutines [24].
LINPACK	Subroutines for the direct solution of linear systems [11].
NSPCG	Subroutines for the iterative solution of linear systems [28].

Chapter 3

RAFT

The three numerical methods described in the previous chapter are implemented on a computer. Most of the code is in FORTRAN 77 except for one subroutine in C++ for integrations over twisted cells in multiple dimensions, ≥ 2 (this has been developed by Doug Moore and Joe Warren of Rice University). This C++ subroutine is only used if the CCM/CMM is the method used to solve the advection-diffusion-reaction equation. The code has been successfully compiled and run on several platforms including SUN, IBM RS/6000, SGI and DEC ALPHA machines.

Overview of RAFT

Problems	RAFT solves Equations 2.1 thru 2.4 along with the differential/algebraic equations (DAEs) $r_i(\{c_n\})$ that describe the reactive flow and transport of groundwater contaminants. Appropriate boundary/initial conditions are chosen depending on the problem. See Chapter 2 for details.
Geometry	RAFT can solve problems in one-, two-, or three-dimensions. The problem domain is restricted to be a cuboid. X-Y-Z cartesian coordinate system is used. X-Y denote the areal extent of the domain and Z is the depth/height.
Input	The input to the code are the physical parameters (like length, porosity, dispersivity, well locations, reaction rate constants, experimental observations etc...) and the parameters to the numerical method (like n_{xp} , n_{yp} , n_{zp} , Δt , error tolerances etc...). See Sections 3.1 and 3.2 for more information on how the input is specified.
Output	Potentials, velocity field, concentration of each component at some time intervals are the typical output from the code. Also some information about mass balance through the course of the simulation is printed out. Flags can be set to demand more output. See Section 3.3 for details.
Units	The code expects all the input data to be in consistent units and does not perform any conversions for output. For example if the length is in meters

Figure 3.1: The cuboid problem domain and the structure of the grid,

and time is in days then one has to specify the hydraulic conductivity in meters/day and expect the potentials to be output in meters.

Reactions RAFT is modular enough that any set of user specified kinetic rate expressions and/or algebraic relations can be incorporated into the code. Appropriate subroutines describing the reaction system should be supplied to RAFT when reactions are present. Section 3.4 explains how this is done. Using a symbolic computational language like MAPLE, the code generation for different reaction systems is automated.

Regression RAFT can be used to estimate the physical and reaction parameters of the process from experimental observations.

3.1 Input Data

Input to RAFT consists primarily of five data files. They are `data`, `columndata`, `welldata`, `datarxn` and `lsqdata`. All are ASCII files.

All the three numerical methods (CCM/CMM, FCT & TVD) use the same input data except for the flag set to indicate the method to be used. Since

the methods are quite different, part of the input data is specific to each method. It is pointed out whenever some input data is specific to one of the methods. Also some input data may have a slightly different interpretation for the three methods.

data This file is a must, to run RAFT. It provides the very basic information about the problem; like the method chosen for solution (CCM/CMM or FCT or TVD), number of components, extent of the domain, boundary conditions, number of wells, etc...

welldata If wells are present in the problem, then this file needs to specify the well locations, flow-rates, etc... This file will not be read if the problem does not have wells (this is specified in **data**).

columndata This file is needed for linear displacement type problems (injection normal to a face). This file specifies the components being injected and their concentrations.

datarxn If the components react, then this file needs to specify the method to be used for solving the ordinary differential equations or the differential-algebraic system (ODES/DAES) and the various constants in the reaction rate expressions. This file will not be read if there are no reactions (specified by a flag in **data**). It is sometimes more efficient to replace some of the ODEs of a reaction system with algebraic, conservation relations (derived by reducing the stoichiometric matrix for the reaction system) and some ODEs with very high rates by equilibrium relations. RAFT is modular enough so that the user needs to modify/supply a subroutine **rxns.f** and/or **preeqrxns.f** for his/her reactions. Section 3.4 describes how this is done and also how **datarxn** is prepared.

lsqdata When RAFT is to be used to estimate parameters from experimental (batch, column or field) observations then this file specifies which method to use for nonlinear regression, which parameters to estimate, etc...

In addition to the above five files there is an "include" file **sizes.h** that sets the maximum sizes for various arrays used in the code.

3.1.1 Description of the file `data`

In what follows here, each input card and the variables it specifies and their significance to the code are described. The names of the variables on each card also appear just before they are entered in to the file `data`. A blank line is needed just before and after each card.

Card # 1: 3 entries.

method An integer. The method chosen for solution. Characteristic method (CCM/CMM) is used when *method* = 0 and the flux-corrected transport method (FCT) is used when *method* = 1. Total variation diminishing method (TVD) is used if *method* = 2.

order An integer. If *method* = 1 in the above then *order* specifies the order of the higher-order fluxes to use. *order* can be 0 or 2 or 4. If *order* = 0 then the high-order correction to the low-order solution is not made. Numerical dispersion can be reduced by using 2nd order or 4th order fluxes. But for some problems this can lead to oscillations and instability. Using *order* = 0 will always give a stable, non-oscillatory but somewhat excessively diffuse solution.

lsqfit An integer flag. If *lsqfit* = 1 then the code will read the file `lsqdata` and estimates the optimal values of selected parameters that minimize/maximize a given objective function. If *lsqfit* = 0 then the file `lsqdata` is not read and no regression is done.

Card # 2: 1 entry.

ncomp An integer. Number of components being modeled.

Card # 3: 3 entries. The problem domain is restricted to be a cuboid, in the X-Y-Z cartesian space. The X-length of the domain is divided into *nxp* (possibly unequal) sections. So if L_x is the extent of the domain in the X-direction and Δx_m is the length of the *m*th section then we need,

$$\sum_{m=1}^{n_x p} \Delta x_m = L_x$$

Similarly the Y-length of the domain is divided into *nyp* sections and the Z-length into *nzp* sections. So the problem domain is divided into *nxp* * *nyp* * *nzp* cells. See Figure 3.1.

- n_{xp}* An integer. The number of sections the X-extent of the domain is divided into.
- n_{yp}* An integer. The number of sections the Y-extent of the domain is divided into.
- n_{zp}* An integer. The number of sections the Z-extent of the domain is divided into.

Card # 4: 4 entries.

- tstart* A real variable. The time at which, the simulation starts.
- tfinal* A real variable. The time at which, the simulation should end.
- Δt A real variable. The time-step size specified by the user. The code may reduce it as required by the numerical method. It may also increase it (see *forcetimestep*).

$$\Delta t \leq \min\{\Delta t_{\text{specified}}, \Delta t_{\text{numerical}}\},$$

where $\Delta t_{\text{numerical}}$ is the time-step size that the numerical method would allow. When wells are being modeled then the maximum time-step size has to be smaller than the time it takes to fill up any injection well-block or empty out a production well-block. Also the successive time-step sizes may be different depending on the injection strategy. RAFT allows for wells to be turned on/off at specified times. Also the injection of components can be turned on/off. The successive time-step sizes are selected such that any on/off switch occurs at the start of a new time-step. This allows the code to exactly model the injection strategy for the problem.

- forcetimestep* An integer flag. Can take values 0 or 1. When sources/sinks are present this flag is used in forcing the code to use a time-step size smaller than the maximum step-size for the problem. If the code finds that $\Delta t_{\text{specified}}$ is smaller than what can be used; it will increase it if *forcetimestep* = 0, but will use $\Delta t_{\text{specified}}$ if *forcetimestep* = 1.

Card # 5: 1 entry.

- restart* An integer flag. Can take values 0 or 1. If this flag is equal to 0, then the initial concentrations for all the components are taken to be uniform. These concentrations *initconc*, are specified in Card # 22. If this flag is equal to 1, then the simulation is started with

the initial (possibly nonuniform) concentrations provided in the file **conc-restart**. See the description of the input file **conc-restart** in Section 3.2.

Card # 6: 6 entries.

donotsolvevel An integer flag. Can take values 0 or 1. If equal to 0 then the code will solve for velocities from the information provided. If equal to 1 then the code will not solve for velocities, but will read the velocities from user-supplied files. The files are **xvelinput**, **yvelinput** and **zvelinput** for X-, Y-, and Z-velocities. See the description of the input files in Section 3.2. Each of these files is necessary only if the number of grid blocks in that direction is greater than 1. If there is only one grid block for a direction then a noflow condition is assumed for that direction.

ipermx An integer flag. Can take values 0 or 1. If this flag is equal to 0, then the hydraulic conductivity in the X-direction (k_x^h) is taken to be uniform. In this case the constant value of $k_x^h \equiv xperm$ is read from Card # 16. If this flag is equal to 1, then k_x^h is taken to be nonuniform. In this case an input file **permx** with the data for k_x^h has to be provided. See Section 3.2.

ipermx An integer flag. Can take values -1 or 0 or 1. If this flag is equal to -1, then the hydraulic conductivity in the Y-direction (k_y^h) is taken to be equal to k_x^h , whether or not k_x^h is uniform. If this flag is equal to 0, then k_y^h is taken to be uniform. In this case the constant value of $k_y^h \equiv yperm$ is read from Card # 16. If this flag is equal to 1, then k_y^h is taken to be nonuniform. In this case an input file **permy** with the data for k_y^h has to be provided. See Section 3.2.

ipermz An integer flag. Can take values -1 or 0 or 1. If this flag is equal to -1, then the hydraulic conductivity in the Z-direction (k_z^h) is taken to be equal to k_x^h , whether or not k_x^h is uniform. If this flag is equal to 0, then k_z^h is taken to be uniform. In this case the constant value of $k_z^h \equiv zperm$ is read from Card # 16. If this flag is equal to 1, then k_z^h is taken to be nonuniform. In this case an input file **permz** with the data for k_z^h has to be provided. See Section 3.2.

iadsorb An integer flag. Can take values 0 or 1. If this flag is equal to 0 then the retardation factor R is taken to be equal to 1.0 for all the

components (that is, none of the components undergo instantaneous adsorption with a linear isotherm). If this flag is equal to 1 then the retardation factor R is not equal to 1.0 for at least one component. Cards # 23, # 24, # 25 and # 26 provide the information about R for each component.

iporos An integer flag. Can take values 0 or 1. If this flag is equal to 0 then the porosity ϕ is taken to be uniform. The value of porosity is read from Card # 14. If this flag is equal to 1 then the porosity ϕ is taken to be nonuniform. An input file **void** to read the porosity from, has to be provided. See Section 3.2.

Card # 7: 1 entry.

ntracetypes An integer. *ntracetypes* is the the number of different R fields. If *method* = 0 then this is also the number of different velocity fields (if two mobile components differ in their retardation factors R , then their velocity fields are different). This has a minimum value of 1 (when all the mobile components have the same R) and a maximum value of *ncomp* (when all components are mobile and have different R).

Card # 8: 1 entry.

nonungrid An integer flag. Can take values 0 or 1. If this flag is equal to 0, then the grid is taken to be uniform i.e., $\Delta x = L_x/nxp$, $\Delta y = L_y/nyp$, $\Delta z = L_z/nzp$. See Figure 3.1. If this flag is equal to 1, then the grid is taken to be nonuniform. An input file **grid** to read the grid from, has to be provided. See Section 3.2.

Card # 9: 3 entries.

dispdiag An integer flag. Can take values 0 or 1. This refers to the form of the dispersion tensor. The dispersion tensor is a 3×3 matrix. If this flag is equal to 1, then the off-diagonal elements of the full dispersion tensor evaluated at a point (Equation 2.4) are not considered (i.e. they are taken to be zero). When this is done the concentration solution resulting from the solution of the elliptic problem (dispersion problem) is completely free of oscillations. If this flag is equal to 0, then the full 3×3 dispersion tensor is used. This can at times lead to small oscillations in the concentration solution.

rxnflag An integer flag. Can take values 0 or 1. If this flag is equal to 0, then no reactions are considered. The input file `datarxn` will not be read. If this flag is equal to 1, then the reactions in the subroutine `rxns.f` (and may be `preeqrxns.f` if equilibration is required; see Section 3.4) will be considered. Also the input file `datarxn` should provide all the rate constants appearing in the reaction rate expressions.

forcevolbal An integer flag. This applies only when $method = 0$. Can take values 0 or 1. This controls whether the code will force the volume-balance criterion or not. If this flag is equal to 1 then the code will try to obtain a velocity solution that obeys the volume-balance property. This flag also controls how the characteristic-integrals are evaluated. An exact integration is necessary for accurate solution and to preserve mass balance. But in multiple dimensions (≥ 2), this is usually very CPU intensive. Depending on the problem, up to 80% of the total CPU time is spent in the exact evaluation of this integral. When a fast solution is desired and accuracy is not too important, one could integrate this approximately. The file `masbal` could be scanned to check whether the error in material balance is tolerable for the application. By choosing this flag to be 1, the characteristic-integral is evaluated exactly. By choosing this flag to be 0, volume-balance criteria is not strictly imposed and the characteristic-integral is evaluated approximately, using Simpson's rule.

This does not apply if $method = 1$ or if $method = 2$.

Card # 10: 1 entry.

timeprint A real variable. This specifies the time-interval between successive printing of concentrations. For example if $timeprint = 1$ day, then the concentrations of all components at the center of all the $n_{xp} \times n_{yp} \times n_{zp}$ grid blocks are printed out to a file `conc` at the end of 1 day, 2 days, 3 days $\dots$ t_{final} . The concentrations at t_{final} are always printed out whether or not t_{final} is an integral multiple of $timeprint$.

Card # 11: 3 entries.

maxtcut An integer. Applies only when $method = 0$. If the Δt specified in Card # 4 is too big, the code will attempt to cut it and use a smaller

time-step size (equal to $0.75 \times \Delta t$). It will continue to cut the time-step until it finds the appropriate time-step size. If the code is cutting the time-step too many times, it is most likely due to an improper specification of the problem (boundary conditions or flow-rates or the lengths etc...). *maxtcut* specifies the maximum number of times the code is allowed to cut the time-step size. Beyond this the code will stop so that the user can reexamine the input to the problem. Alternatively *maxtcut* can be increased.

This does not apply to *method* = 1 and *method* = 2.

masbalcomp An integer. The code prints out material balance information for the component *masbalcomp* to the file masbal. *masbalcomp* is the number (identity) of the component (components are numbered 1 thru *ncomp*) for which this information is reported. See Section 3.2 to find out what this file masbal contains.

ntracksteps An integer. Applies only when *method* = 0. The movement along the streamlines is usually computed with a forward Euler scheme involving multiple steps. *ntracksteps* sets the number of these steps. A large number of steps could be used to increase the accuracy but at the cost of CPU time. When *forcevolbal* = 1, setting *ntracksteps* equal to 1 would help the velocity solution to converge faster. If *forcevolbal* = 0 a large number of *ntracksteps* could be used.

This does not apply to *method* = 1 and *method* = 2.

Card # 12: 4 entries.

maxiterc An integer. This sets the maximum number of iterations allowed for convergence of the concentration problem, when the zero dispersive flux condition is used at outflow boundaries. The smaller the dispersion fewer will be the number of iterations required. When dispersion is zero, there is no linear system to solve and so *maxiterc* is not used.

epsc A real variable. The residual in the preconditioned conjugate gradient method is reduced by this factor.

maxiterp An integer. Applies only when *method* = 0. This is the maximum number of iterations allowed for convergence of the velocity problem. When *method* = 0, the velocity problem involves the solution of a nonlinear system when *forcevolbal* = 1. When R_i or ϕ is nonuniform, each iteration takes more CPU time and also the convergence is slow.

In cases of severe heterogeneity in porosity ϕ or retardation R_i , the code may have problems converging to the specified residual. It may help in such cases to (a) increase the tolerance *maxvolerror*, the fourth entry on Card # 12 or/and, (b) set the fourth entry *ntracksteps* on Card # 11 to 1, to achieve convergence.

When the desired convergence is not achieved in *maxiterp* iterations, the code will complain,

Velocity Problem Did Not Converge To the Specified Residual

but will continue to run.

maxvolerror A real variable. Applies only when *method* = 0. The velocity solution is based on a volume-balance scheme. If the entry *forcevolbal* in Card # 9 is set equal to 1 then the code will try to conserve fluid volume. *maxvolerror* is the biggest relative error in volume that is acceptable at convergence. If the relative error in volumes, is greater than *maxvolerror* then the code will warn,

Volumes very different,

followed by the indices of the grid block for which this happened, the correct volume, and the computed volume. This will always happen for those grid blocks associated with wells, the difference being the volume of fluid entering/leaving during this time-step (equal to $\hat{q}_{wk}\Delta t$). This warning may also appear for other grid blocks if *forcevolbal* in Card # 9 is set equal to 0 (thereby not requiring the volumes to match). But if *forcevolbal* is set equal to 1 this warning should not appear for grid blocks not associated with wells, provided of course that the method converged.

Card # 13: 3 entries. See Figure 3.1.

L_x A real variable. This is the X-extent of the problem domain.

L_y A real variable. This is the Y-extent of the problem domain.

L_z A real variable. This is the Z-extent of the problem domain.

Card # 14: 1 entry.

porosity A real variable. If the 6th entry on Card # 6 *iporos* is equal to 0 then the porosity is taken to be uniform. In this case *porosity* is the

constant porosity of the problem domain. If *iporos* is equal to 1 then this value is ignored and the field of porosity values is read from the file `void`.

Card # 15: 1 entry.

M A real variable. This is not used by this version of the code. *M* is always set equal to 1.0.

Card # 16: 3 entries.

xperm A real variable. If the hydraulic conductivity in the X-direction (k_x^h) is taken to be uniform (by setting *ipermx* equal to 0 in Card # 6, then *xperm* gives the value of this constant.

yperm A real variable. If the hydraulic conductivity in the Y-direction (k_y^h) is taken to be uniform (by setting *ipermx* equal to 0 in Card # 6, then *yperm* gives the value of this constant.

zperm A real variable. If the hydraulic conductivity in the Z-direction (k_z^h) is taken to be uniform (by setting *ipermz* equal to 0 in Card # 6, then *zperm* gives the value of this constant.

Card # 17: 3 entries.

d_m A real variable. This is the molecular diffusivity *d_m* used in Equation 2.4.

α_l A real variable. This is the longitudinal dispersivity used in Equation 2.4.

α_t A real variable. This is the transverse dispersivity used in Equation 2.4

Card # 18: 7 entries. Flags. See Figure 3.1. The first six flags in this Card specify the type of boundary condition for the velocity problem, used on each of the six faces. If the entry is 1, a constant potential boundary condition (Equation 2.5b) is used. If the entry is 0, a noflow boundary condition (Equation 2.5a) is used.

pbcface(1,1) Boundary condition at $X = 0$.

pbcface(1,2) Boundary condition at $X = L_x$.

pbcface(2,1) Boundary condition at $Y = 0$.

pbcfac(2,2) Boundary condition at $Y = L_y$.

pbcfac(3,1) Boundary condition at $Z = 0$.

pbcfac(3,2) Boundary condition at $Z = L_z$.

If *donotsolvevel* = 1 then *pbcfac* is set equal to 1 for all boundaries. The 7th entry *cbc* specifies the boundary condition to be used for the concentration (dispersion) problem at outflow boundaries (Danckwerts condition, which translates to zero dispersive flux condition, is imposed at inflow boundaries). When *cbc* = 0 then a zero dispersive flux condition is used over the outflow boundaries. When *cbc* = 1 then the dispersive flux across the outflow boundary is approximated to be equal to the flux just inside that boundary. This leads to a nonsymmetric linear system but yields a better solution whenever dispersive flux is not negligible.

Card # 19: 6 entries. See Figure 3.1. If a constant potential boundary condition is specified for a face on Card # 18, then the entries in this card give the value of this constant. If a noflow condition is specified then the corresponding entry in this card will be ignored.

Card # 20: 5 entries.

presflag An integer. A flag. If it is equal to 1, then the potentials at the center of each grid block are printed out to a file potentials at the end of the velocity problem.

If *method* = 0, then the potentials will be printed out for each trace-type. Since the velocity solution depends on the time-step size Δt , new velocities are solved every time Δt changes. So if the time-step sizes are nonuniform then velocity problem may be solved many times and so can generate a lot of output to this file. This printing is suppressed if this flag is set equal to 0.

velflag An integer. A flag. If this is equal to 1, then the X-, Y- and Z-velocities at the center of each grid block are printed out to files xvel, yvel and zvel respectively. Again if *method* = 0 and if the time-step sizes are nonuniform, a lot of output may be generated to these files. Printing is suppressed if this flag is set to 0.

outputflag An integer flag. The code prints out a lot of miscellaneous information as the simulation progresses. This provides a useful check on whether the simulation is progressing correctly or not. When *outputflag* = 1,

this information is printed to the screen. But for long simulations the output file can get large and so the printing may be turned off by setting *outputflag* = 0. Setting *outputflag* = -1 will cause the mass balance information also to not to be printed to the file masbal (see Section 3.3). This is useful for a simulation with a large number of time-steps.

Card # 21: 2 entries.

- iwells* An integer. This is the total number of injection wells being modeled.
- pwells* An integer. This is the total number of production wells being modeled.

Card # 22: *ncomp* entries.

- initconc* This is an array that specifies the initial concentrations for *ncomp* components. If the entry *restart* on Card # 5 is equal to 0 then the initial concentrations of all components are taken to be uniform. In this case the entries in this card give the constant concentrations for components 1 thru *ncomp*. If *restart* is equal to 1 then these entries will be read but ignored.

Card # 23: *ntracetypes* entries.

- stick* Integers. Takes values 0 or 1. If this flag is equal to 0, then the components under this tracetype do not adsorb (that is $R_i \equiv 1.0$). If it is equal to 1, then the components under this tracetype are taken to adsorb. If *iadsorb* (on Card # 6) is set equal to 0, all the entries on this card will be forced to 0 (i.e., no adsorption). When *method* = 0, then the components under a given "tracetype" have the same velocity field (that is, same R_i since $\underline{k}^h$, ϕ and Δt are the same for all components).

Card # 24: *ntracetypes* entries.

- nustick* Real variables. This is an array that specifies the value of R_i for each tracetype, when *iadsorb* and *stick*(tracetype) are equal to 1. If *nustick* is different from 1.0, it gives the value of the "uniform" R_i for that tracetype.

$$R_i \equiv \text{nustick}(\text{tracetype})$$

If this value is equal to 1.0, R_i for that tracetype is taken to be "nonuniform" and a file `adsorb` has to be supplied with all the $n_{xp} * n_{yp} * n_{zp}$ values of R_i for that tracetype. If either $iadsorb$ or $stick(tracetype)$ is equal to zero then R_i is identically equal to unity for the components under that tracetype.

Card # 25: $n_{tracetypes} + 1$ entries.

ntracecomp Integers. *ntracecomp* is an array (of length $n_{tracetypes}$) that gives the number of components that fall under each tracetype. Can take values 1 thru n_{comp} .

nimmobilecomp Integer. The number of immobile components. We note that,

$$\sum_{i=1}^{n_{tracetypes}} n_{tracecomp}(i) + n_{immobilecomp} = n_{comp}$$

Card # 26: $n_{tracetypes} + 1$ lines.

compId Components are numbered from 1 to n_{comp} . The "numbers" (identification) for components under a given tracetype are listed here; one card for each tracetype.

immobilecomp The "numbers" (identification) for the immobile components.

Card # 27: n_{comp} lines and two entries per line. The code needs to know what the minimum and maximum possible concentrations are for each component. These may be imposed while the reaction expressions are integrated and also to check whether the concentrations remain bounded as the simulation progresses.

minconc A real variable. The minimum possible concentration for a component.

maxconc A real variable. The maximum possible concentration for a component.

Card # 28: 4 entries.

nhistlocs Integer. In addition to printing out the entire concentration profiles for each component at selected time intervals through *timeprint* (Card # 10), the code can be made to print out the history of concentrations at selected spatial locations. *nhistlocs* is the number of such locations in the grid domain where such a history is desired.

bkthru Integer. It is often of interest to follow the average concentration breaking through a given cross section. When *bkthru* = 1, then the breakthrough concentration across the YZ planes (*X* = constant) is printed out to the file histconc. If a component is mobile then its breakthrough concentration at a location *x_b* is computed as,

$$\frac{\int_{y=0}^{y=L_y} \int_{z=0}^{z=L_z} C(x_b, y, z) V_x(x_b, y, z) dy dz}{\int_{y=0}^{y=L_y} \int_{z=0}^{z=L_z} V_x(x_b, y, z) dy dz}.$$

If the component is immobile then an arithmetic average is computed.

nhistcomp Integer. The number of components for which history is desired.

histime A real variable. Similar to *timeprint* (Card # 10), but for printing concentration history. This is the time interval between successive printings of concentration history for the selected components at the selected locations.

Card # 29: *histlocs*. Array of dimensions (*nhistlocs*,3). Integers. *nhistlocs* lines, 3 entries per line. Each line contains the (x,y,z) grid coordinates of one of the *nhistlocs* points in the grid. When *bkthru* = 1, only the x-grid location (the first entry in each line) is used since the breakthrough is measured across the YZ planes.

Card # 30: *histcomp*. Array of length *nhistcomp*. Integers. The "numbers" (identities) of the components for which history is desired.

3.1.2 Description of the file columndata

If the problem features external injection normal to a face then this file will be read. In what follows here, each input card and the variables it specifies and their significance to the code are described. Again the units have to be consistent. The names of the variables on each card also appear just before they are entered in to the file. A blank line is needed just before and after each card.

Card # 1: 1 entry.

injfaceflag An integer flag. Can take values 0 or 1. If this flag is set to 0, then the problem being solved should have wells. Alternatively, If max(*iwells*, *pwells*) on Card # 21 is greater than 0 then *injfaceflag* is forced to be zero and the file columndata is not read. When *iwells* + *pwells* = 0 and *injfaceflag* = 1 then a linear displacement type problem is solved. Injection is always normal to *X* = 0 face. See Figure 3.1.

Card # 2: 2 entries.

flowflag An integer flag. Can take values 0 or 1. If this is set to 1, then the total flow-rate into the domain is forced to be equal to *flowrate*. The potential drop (between $x = 0$ and $x = L_x$) specified (in Card # 19) across the flow is adjusted to achieve this. If this is set to 0, then the flow-rate into the domain is determined by the potential drop specified in Card # 19.

flowrate A real variable. When *flowflag* is set to 1, then the total flow-rate into the domain is forced to be equal to this value. Otherwise this is ignored.

Card # 3: *ncomp* entries.

Finject Integers. Flags. *Finject* is an array (of length *ncomp*) that specifies for each component whether it is being injected or not. If a component *comp* is being injected at the face $X = 0$, then *Finject(comp)* is set equal to 1. Otherwise *Finject(comp)* is set equal to zero.

Card # 4: *ncomp* entries.

Fswitchcomp Integers. The injection scheme for the components is very flexible. The injection can be turned off/on at any time. The injection concentration can change with time. Also each component can have its own injection strategy. So the injection of the components is modeled as a series of (possibly irregularly spaced) pulses of concentration (possibly different for each pulse). *Fswitchcomp* is an array that gives the number of such pulses for each component being injected.

Card # 5 1 entry.

nonuniforminj An integer. A flag. Takes values 0 or 1. If equal to 0 then the injection on the $X = 0$ face is taken to be uniform. If equal to 1 then the injection is nonuniform and the following card provides the part of the $X = 0$ face over which injection takes place. This applies only when *method* = 1 or *method* = 2. For *method* = 0, the injection is always uniform over the face.

Card # 6 4 entries. Integers. *injylow*, *injyhigh*, *injzlow*, *injzhigh*. Apply only when *method* = 1 or *method* = 2 and *nonuniforminj* = 1.

If *nonuniforminj* is not equal to 0 in the above card then the grid coordinates of the rectangular area on the $X = 0$ face used for injection should be given here.

$$1 \leq injy_{low}, injy_{high} \leq nyp$$

$$1 \leq injz_{low}, injz_{high} \leq nzp$$

If the entire face is used for injection then $injury_{low} = 1$, $injury_{high} = nyp$, $injz_{low} = 1$, and $injz_{high} = nzp$. This is equivalent to setting *nonuniforminj* to 0.

Cards # 7: $\sum_{comp=1}^{ncomp} F_{switchcomp}(comp)$ lines. For those components that are injected, the injection strategy is specified here. Injection strategy consists of the number of pulses specified on Card # 4, the start/end times for each pulse and the concentration of each pulse. So there is one line with three entries for each pulse for each injected component.

Fcompstarttime A real variable. The starting time of a pulse for an injected component.

Fcompendtime A real variable. The ending time of this pulse for this injected component.

Foriginjconc A real variable. The concentration of this pulse for this injected component.

3.1.3 Description of the file welldata

This file is read only if $\max(iwells, pwells)$ on Card # 21 in the file data is greater than 0. In what follows here, each input card and the variables it specifies and their significance to the code are described. Again the units have to be consistent. A brief description of these variables also appears at the top of the file welldata for quick reference during its preparation. The first input card starts at/after the 45th line of the file.

Card # 1: 2 entries.

implicitrates An integer flag. Can take values 0 or 1. If equal to 1 then the splitting of the flow out/into a well among the layers is modeled implicitly. If equal to 0 then the flow in/out a well to the k th layer

$(\hat{q}_{wk})$ is approximated as,

$$\lambda_{wk} = \Delta z_k (k_x^h k_y^h)^{1/2}, \hat{q}_{wk} = Q_w \frac{\lambda_{wk}}{\sum_{m=k_w^{bot}}^{k_w^{top}} \lambda_{wm}}.$$

This neglects the coupling of the well-blocks. Q_w is the total flow rate in/out of the well. Note that the time-step size should be such that

$$\Delta t \leq \text{Min} \left[\frac{\Delta x_w^m \Delta y_w^j \Delta z_w^k \phi}{|\hat{q}_{wk}|} \right],$$

where (m, j, k) is a well-block.

reinjection An integer flag. This specifies whether any production well(s) reinject into some injection well(s). If this flag is equal to 1, then such a reinjection will take place according to the specifications that follow later in this file. This will affect the injection concentration of a component at the injection well, *only* when there is no direct external injection of that component at that injection well. When external injection is on, then the injection concentration is not changed due to reinjection.

The following cards are required for each injection well.

Card: 4 entries.

lociwell 4 Integers. This is an array that specifies the location of this injection well in the X-Y-Z cartesian grid space. All wells are vertical and occupy a single grid block in the X-Y plane. The screen depth/height may vary from 1 to nzp grid blocks. So four numbers specify the location of the well, $(i_w, j_w, \{k_w^1 \cdots k_w^2\})$.

i_w : X-location of the well. ≥ 1 and $\leq nxp$.

j_w : Y-location of the well. ≥ 1 and $\leq nyp$.

k_w^1 : Bottom Z-location of the screen. ≥ 1 and $\leq nzp$.

k_w^2 : Top Z-location of the screen. ≥ 1 and $\leq nzp$.

Card: 1 entry.

irw A real variable. This is the radius of the well.

Card: 1 entry.

nswitchiwell An integer. The number of times this injection well is switched on-off (i.e., pulsed).

Card: *nswitchiwell* lines. These cards specify the pulsing strategy for this injection well. This consists of the triplet of numbers, { *iwellstarttime*, *iwellendtime*, *origtotiqrate* } for each pulse.

iwellstarttime A real variable. The starting time of this pulse for this injection well.

iwellendtime A real variable. The time when this pulse for this injection well ends.

origtotiqrate A real variable. The flow-rate into this injection well during this time.

Card: *ncomp* entries.

Winject Integers. Flags. *Winject* is an array that specifies for each component whether it is being injected at this well or not. If a component *comp* is being injected in a well *iwell* then *Winject(comp,iwell)* is set equal to 1. Otherwise *Winject(comp,iwell)* is set equal to zero.

Card: *ncomp* entries.

Wswitchcomp Integers. The injection scheme for the components is very flexible. The injection can be turned off/on at any time. The injection concentration can change with time. Also each component can have its own injection strategy. So the injection of the components is modeled as a series of (possibly irregularly spaced) pulses of concentration (possibly different for each pulse). *Wswitchcomp* is an array that gives the number of such pulses for each component being injected at this well.

Card: $\sum_{comp=1}^{ncomp}$ *Wswitchcomp(comp,iwell)* lines. For those components that are injected, the injection strategy is specified here. Injection strategy consists of the number of pulses specified before, the start/end times for each pulse and the concentration of each pulse. So there is one card with three entries for each pulse for each injected component at this well. Note that no components are injected when the injection well is OFF.

Wcompstarttime Real variable. The starting time of a pulse for an injected component at this well.

Wcompendtime Real variable. The ending time of this pulse for this injected component at this well.

Woriginjconc Real variable. The concentration of this pulse for this injected component at this well.

The following cards are required for each production well.

Card: 4 entries.

locpwell 4 Integers. This is an array that specifies the location of this production well in the X-Y-Z cartesian grid space. All wells are vertical and occupy a single grid block in the X-Y plane. The screen depth/height may vary from 1 to *nzp* grid blocks. So four numbers specify the location of the well, $(i_w, j_w, \{k_w^1 \cdots k_w^2\})$.

i_w : X-location of the well. ≥ 1 and $\leq nxp$.

j_w : Y-location of the well. ≥ 1 and $\leq nyp$.

k_w^1 : Bottom Z-location of the screen. ≥ 1 and $\leq nzp$.

k_w^2 : Top Z-location of the well. ≥ 1 and $\leq nzp$.

Card: 1 entry.

prw A real variable. This is the radius of the well.

Card: 1 entry.

nswitchpwell An integer. The number of times this production well is switched on-off (i.e., pulsed).

Cards: *nswitchpwell* cards. These cards specify the pulsing strategy for this production well. This consists of the triplet of numbers, $\{ pwellstarttime, pwellendtime, origtotpqrte \}$ for each pulse.

pwellstarttime A real variable. The starting time of this pulse for this production well.

pwellendtime A real variable. The time when this pulse for this production well ends.

origtotpqrte A real variable. The flow-rate out of this production well during this time.

Card: 1 entry.

p2iwellmap An integer. This is the "number" (identification) of the injection well into which this production well reinjects, when the flag *reinjection* is set equal to 1 on Card # 1. If *reinjection* is set equal 0, this entry will be ignored.

3.1.4 Description of the file `sizes.h`

This file sets the maximum sizes for various arrays used in the code, thereby determining the memory requirements for running RAFT. If the data files require larger sizes than what is allocated here, the code may stop or produce incorrect results. To reduce the memory requirements it also sets the flag for the method to be used and accordingly allocates memory for the arrays. The code should be recompiled whenever `sizes.h` is modified.

- methodflag* The numerical method to be used in solving the problem. The flag *method* in Card # 1 of file `data` should match with this. Otherwise the code will stop and ask the user to recompile with *methodflag* set the same as *method* or change the file `data` so that *method* = *methodflag*.
- maxwells* This should be greater than or equal to the maximum of *iwells* and *pwells*. $\text{maxwells} \geq \max\{\text{iwells}, \text{pwells}\}$.
- maxntracetypes* This should be greater than or equal to *ntracetypes*. $\text{maxntracetypes} \geq \text{ntracetypes}$.
- maxpulses* This should be greater than or equal to the maximum number of pulses being considered for any well or component.
- If $(\max(\text{iwells}, \text{pwells}))=0$, $\text{maxpulses} \geq \max\{\text{Fnswitchcomp}(1..\text{ncomp})\}$.
If $(\max(\text{iwells}, \text{pwells}))>0$, $\text{maxpulses} \geq \max\{\text{nswitchiwell}(1..\text{iwells}), \text{nswitchpwell}(1..\text{pwells}), \text{Wnswitchcomp}(1..\text{ncomp}, 1..\text{iwells})\}$
- maxtsteps* The maximum number of time-steps that may be needed. At a minimum,
- $\text{maxtsteps} > (\text{tfinal} - \text{tstart})/\Delta t$
- maxtsteps* may have to be bigger depending on whether the code will cut the time-step size. Also the injection strategy will affect the number of time-steps needed.
- maxncomp* The maximum number of components being considered. $\text{maxncomp} \geq \text{ncomp}$.
- inxp* Maximum number of grid blocks along the X-direction. $\text{inxp} \geq \text{nxp}$.
- inyp* Maximum number of grid blocks along the Y-direction. $\text{inyp} \geq \text{nyp}$.
- inzp* Maximum number of grid blocks along the Z-direction. $\text{inzp} \geq \text{nzp}$.

maxnrnconsts The maximum number of constants to be used in the reaction rate expressions.

maxnw The size of the real work array used by **nspcg** (iterative solver for the velocity problem) If the size specified here is insufficient, code will exit with an error message,

```
*****  
***** f a t a l   e r r o r *****  
*****  
  
insufficient real workspace  
  
*****
```

nspcg estimates the size of *maxnw* and prints to the standard output,

minimum NW needed : xxxxxxxx

When *method* = 1 or *method* = 2 then this value for NW is just sufficient. When *method* = 0, this size may not be sufficient (perhaps a bug in **nspcg** ?). If not *maxnw* should be further increased. Alternatively, a large value of *maxnw* could be used for all problems so that it will always be sufficient. But this will increase the memory requirements.

maxinw The size of the integer work array used by **nspcg** (iterative solver for the velocity problem). If the size specified here is insufficient, code will exit. An error message like above will appear. Again the estimate for *maxinw* from RAFT can be used as a guide to set this.

maxnhistlocs The maximum number of locations at which a history of concentration is desired.

maxnob The maximum number of experimental observations need to be considered for regression.

maxnfitparams The maximum number of parameters to estimate from the experimental observations. This is never greater than $2 + \text{maxntracetypes} + \text{maxnrnconsts}$.

MMAXNUMT This is the maximum of number of terms occurring in any of the non-linear equations for solution with the package INTBIS. If INTBIS is not being used then this can be set to unity.

MMAXLS This the parameter "MMAXLS" of the package INTBIS. Set it to unity (to save memory) if INTBIS is will not be used during the simulation.

MMAXDP This the parameter "MMAXDP" of the package INTBIS. Set it to unity (to save memory) if INTBIS is will not be used during the simulation.

A program `sizes.f` uses `sizes.h` to compute the sizes of various work arrays and writes it to the file `usesizes.h`. `usesizes.h` should be regenerated whenever `sizes.h` changes.

3.2 Input Files

Additional input files may be necessary to run RAFT depending on the flags set in the file `data`.

adsorb If *iadsorb* is equal to 1 and *stick(tracetype)* is equal to 1 and *nu-stick(tracetype)* is equal to 1.0 then the code will look for this file to read the nonuniform R_i data for this tracetype. The FORTRAN statement for reading is,

```
read (adsorbunit,*) (((sorp(i,j,k,tracetype),i=1,nxp),  
                      j=1,nyp),k=1,nzp)
```

If several tracetypes have nonuniform R_i then each array of data has to go into this file one after the other, $1 \cdots ntracetypes$.

conc-restart If *restart* is set equal to 1 then this file has to supply the concentrations of all components at that time. The following FORTRAN statements read the file.

```
read (concrestartunit,*) (((((csol(i,j,k,comp),i=1,nxp),  
                             j=1,nyp),k=1,nzp),comp=1,ncomp)
```

grid If *nonungrid* is set equal to 1 (implying that the grid is nonuniform) then this file with the locations of the X-, Y- and Z-grid planes has to be

provided. This file is necessary even if only one of them is nonuniform. The structure of this file is,

$$\begin{aligned} &0.0d0, \Delta x_1, \Delta x_1 + \Delta x_2, \Delta x_1 + \Delta x_2 + \Delta x_3, \dots L_x \\ &0.0d0, \Delta y_1, \Delta y_1 + \Delta y_2, \Delta y_1 + \Delta y_2 + \Delta y_3, \dots L_y \\ &0.0d0, \Delta z_1, \Delta z_1 + \Delta z_2, \Delta z_1 + \Delta z_2 + \Delta z_3, \dots L_z \end{aligned}$$

So there are $n_x p + 1 + n_y p + 1 + n_z p + 1$ entries in the file. The following FORTRAN statements read the file.

```
read (gridunit,*) (xp(i),i=1,nxp+1)
read (gridunit,*) (yp(i),i=1,nyp+1)
read (gridunit,*) (zp(i),i=1,nzp+1)
```

permx

This file will be necessary if *ipermx* is set equal to 1 (implying that k_x^h is heterogeneous). The value of k_x^h in each of the $n_x p * n_y p * n_z p$ grid blocks should go into this file. This file will be read by the following FORTRAN statement.

```
read (permxunit,*) (((permx(i,j,k),i=1,nxp),j=1,nyp),
                    k=1,nzp)
```

permy

This file will be necessary if *ipermy* is set equal to 1 (implying that k_y^h is heterogeneous). This file is prepared the same way as the file **permx**.

permz

This file will be necessary if *ipermz* is set equal to 1 (implying that k_z^h is heterogeneous). This file is prepared the same way as the file **permx**.

void

If *iporos* is set equal to 1 then the code will look for this file to read porosity ϕ from. This file will be read by the following FORTRAN statement.

```
read (voidunit,*) (((poros(i,j,k),i=1,nxp),j=1,nyp),
                    k=1,nzp)
```

xvelinput

When *donotsolvevel* = 1, then this file should supply the X-velocities. When *method* = 0 then the file is read in by the following code.

```
read (inputxvelunit,*) (((velx(i,j,k,tracetype),i=1,2*nxp+1),
                        j=1,2*nyp+1),k=1,2*nzp+1),tracetype=1,ntracetypes)
```

If *method* = 1 or 2 then the following code is used since the velocities are the same for all tracetypes.

```
read (inputxvelunit,*) (((velx(i,j,k),i=1,nxp),
                        j=1,nyp),k=1,nzp)
```

yvelinput

Similar to **xvelinput**.

zvelinput

Similar to **zvelinput**.

3.3 Output Files

In addition to the velocities and concentrations of all the components RAFT outputs other information helpful in tracking the progress of the simulation and in interpreting the results. If some of this output is not necessary they can be turned off by setting appropriate flags in the file **data**.

Cprintimes

The "times" at which the concentrations were output, are printed out to this file. These times are determined by *timeprint* on Card # 10 in the file **data**. Usually, these times are equal to,

$$tstart + timeprint, tstart + 2 * timeprint, \dots tfinal.$$

The concentrations at *tfinal* are always printed out whether or not *tfinal* is an integral multiple of *timeprint*.

conc

The values of concentration at the center of each grid block for each component are printed out to this file, at a specified time-interval *timeprint* according to the FORTRAN statement,

```
do 10 comp = 1, ncomp
    write (concunit,25) (((csol(i,j,k,comp),
&                        i=1,nxp),j=1,nyp),k=1,nzp)
10    continue
25    format (6(e20.12e3))
```

- debug A lot of miscellaneous information may be printed to this file, if the code detects unsatisfactory local mass balance (that is mass balance for each grid block), post-processing etc... But the code will continue to run. This is primarily of use in debugging.
- histconc If *nhistlocs* > 0 then a history of concentration at selected locations is printed to this file. At each time-step *nhistcomp* lines will be printed. Each line will have 1 + *nhistlocs* entries. The first quantity is the time at which it is being printed and the rest are the concentration values for that component at each of the *nhistlocs* locations.
- masbal The information about mass balance as the simulation proceeds is printed to this file after every time-step. The component for which mass balance is reported is *masbalcomp* on Card # 11 in the file `data`. Set *outputflag* = -1 the file `data` if not interested in this information. One line is printed for each time-step. Its contents are the following.
- time The time at the end of this time-step.
 - consume The mass of this component that reacted during this time-step.
 - totconsume The total mass of this component that reacted since the time *tstart*.
 - totbcflowin The total mass of this component that entered the domain from across the boundaries, since the time *tstart*.
 - totbcflowout The total mass of this component that left the domain from across the boundaries, since the time *tstart*.
 - totcflowin The total mass of this component that entered the domain, either through injection wells or through flow across the boundaries, since the time *tstart*.
 - totcflowout The total mass of this component that left the domain, either through production wells or flow across the boundaries, since the time *tstart*.
 - cpresent The total mass of this component that is present in the domain at this time.
 - diffcbal The differential mass balance. Mass balance should hold at every time-step. For a time-step, *diffcbal* is computed as,

$\text{diffcbal} = \text{Mass that left the domain} - \text{Mass that entered the domain} + \text{Mass that disappeared due to reaction} + \text{Change in the Mass that is present in the domain.}$

diffcbal will be equal to zero if mass is preserved in the simulation. When $\text{method} = 0$, the mass that entered/left the domain across the boundaries is sometimes computed approximately. Also the volume balance is only satisfied to the tolerance specified in maxvolerror (4th entry on Card # 12 in the file `data`). This may lead to small nonzero values for diffcbal when $\text{method} = 0$. Even larger errors may show up if $\text{forcevolbal} = 0$ in Card # 9 of the file `data`.

cumcbal The cumulative mass balance. Mass balance should also hold over the entire length of simulation since the time $tstart$. At any time,

$\text{cumcbal} = \text{Total mass that left the domain since the time } tstart - \text{Total mass that entered the domain since the time } tstart + \text{Total mass that disappeared due to reaction since the time } tstart + \text{Total mass that is currently present in the domain} - \text{Total mass present at the initial time } tstart.$

cumcbal will be equal to zero if mass is preserved. In the code, the mass that entered/left the domain is sometimes computed approximately. This may lead to small nonzero values for cumcbal when $\text{method} = 0$. Even larger errors may show up if $\text{forcevolbal} = 0$ in Card # 9 of the file `data`.

cbarbal Applies only when $\text{method} = 0$. This is a check on the integrations for the characteristic-integral. If the volume balance is achieved and if the characteristic-integral is evaluated exactly and the linear dispersion problem converges satisfactorily then cbarbal , diffcbal and cumcbal will be zero.

noscmass The code forces the concentrations of each component to lie between the maximum and minimum values specified in the file `data`. Usually the concentrations do not overshoot the maximum and undershoot the minimum possible values. For $\text{method} = 0$, overshoot/undershoot may occur when volumes do not match or when $\text{dispdiag} = 0$. For $\text{method} = 1$, this can

happen if the method becomes unstable or when $dispdiag = 0$. For $method = 2$, this can happen when $dispdiag = 0$. $noscmass$ is the amount of mass of the component *masbalcomp* gained by forcing the concentration to be equal to the minimum value when it actually undershoots.

poscmass Similar to $noscmass$, this is the amount of mass of the component *masbalcomp* lost by forcing the concentration to be equal to the maximum value when it actually overshoots.

potentials See *presflag* on Card # 20 in the file **data**. The values of potential at the center of each grid block are printed out to this file according to the FORTRAN statement,

```

                write (presunit,25) (((psol1(i,j,k),i=1,nxp),
&                j=1,nyp),k=1,nzp)
25      format (6(e20.12e3))

```

When $method = 0$ then this will be done for each tracetype.

stdout If *outputflag* = 1 then as the simulation proceeds, the code will echo the progress to the screen (or to a file if it is redirected). Most of the input data is immediately echoed, so the user can check whether the input data has been read in correctly or not. Also the run-time errors if any will be directed here. For long simulations this file can get large. If not interested in this information *outputflag* can be set equal to 0.

wellrates The flow-rates $\hat{q}_{wk}$ into each well-block for each well are printed out to this file. If $\max(iwells, pwells) = 0$ and *flowflag* = 0 (Card # 2 in the file **columnndata**) then the flow-rate into the domain at $x = 0$ is also printed to this file. If $\max(iwells, pwells) = 0$ and *flowflag* = 1 then the inlet ($x = 0$) potential is printed to this file.

xvel, yvel, zvel See *velflag* on Card # 20 in the file **data**. The X-, Y-, Z-velocities are printed out to the files **xvel**, **yvel** and **zvel** respectively. When $method = 0$ then for each tracetype the following FORTRAN statements are used in writing.

```

write (xvelunit,25) (((velx(i,j,k,tracetype),

```



```

&      i=1,2*nxp+1),j=1,2*nyp+1),k=1,2*nzp+1)
      write (yvelunit,25) (((velx(i,j,k,tracetype),
&      i=1,2*nxp+1),j=1,2*nyp+1),k=1,2*nzp+1)
      write (zvelunit,25) (((velz(i,j,k,tracetype),
&      i=1,2*nxp+1),j=1,2*nyp+1),k=1,2*nzp+1)
25      format (6(e20.12e3))

```

All three velocities are defined at the same locations on the grid. The locations for these velocities are clearly the intersection points of the tensor grid formed by $2 * nxp + 1$ x-grid planes, $2 * nyp + 1$ y-grid planes and $2 * nzp + 1$ z-grid planes. Of the $2 * nxp + 1$ x-grid planes, $nxp + 1$ planes are at the edges of the blocks and nxp are at the centers of the blocks. Similarly the y- and z-grid planes are located. Figure 3.2A shows the locations (marked "o") on a typical X-Y grid where the X-, and Y-velocities are defined.

When *method* = 1 or *method* = 2, following FORTRAN statements are used.

```

      write (xvelunit,25) (((velx(i,j,k),
&      i=1,nxp+1),j=1,nyp),k=1,nzp)
      write (yvelunit,25) (((velx(i,j,k),
&      i=1,nxp),j=1,nyp+1),k=1,nzp)
      write (zvelunit,25) (((velz(i,j,k),
&      i=1,nxp),j=1,nyp),k=1,nzp+1)
25      format (6(e20.12e3))

```

The x-velocities are printed at the centers of YZ-faces of every grid block. Similarly the y-velocities and z-velocities. Figure 3.2B shows the locations on a typical X-Y grid where the X-, and Y-velocities are defined. The X-velocities are defined at locations marked "x". The Y-velocities are defined at locations marked "y".

3.4 Reactions

The use of RAFT would be quite limited if the set of reactions $r_i(\{c_n\})$ cannot be easily incorporated into the code. Different well configurations and injection schemes are handled by RAFT through the input data. But to solve a problem with a different

Figure 3.2: A. X-, Y-velocities are defined at the locations marked "o", for $method = 0$. B. X-velocity is defined at locations marked "x" and Y-velocity at "y" for $method = 1$ or $method = 2$.

set of reactions, the user has to provide a few subroutines that describe the reaction system.

3.4.1 Formulation of the reaction system

The reaction system is in general a differential-algebraic equation system (DAE system); differential equations resulting from rate equations and the algebraic equations coming from conservation laws or equilibrium relations. Differential-algebraic systems are characterized by an index. For a definition of the index of a DAE system the user is referred to [15, 8]. The index of a reaction system may be loosely defined as the number of times the equilibrium relations need to be differentiated to convert the system to an all ODE system. RAFT has so far been only used in the solution of unit index or zero index DAE systems. Zero index DAE system is an ODE system. Higher index (index > 1) systems are difficult to solve. For our purposes the most general system can be expressed as,

$$\underline{\underline{M}} \frac{d\underline{Y}}{dt} = \underline{F}(\frac{d\underline{Y}}{dt}, \underline{Y}, t), \quad (3.1)$$

where $\underline{Y}$ is the vector (of length $ncomp$) of component concentrations. $\underline{F}$ is in general a function of $\frac{d\underline{Y}}{dt}$, $\underline{Y}$ and t . For most problems of our interest $\underline{F}$ is only a function of $\underline{Y}$. $\underline{\underline{M}}$ is a constant matrix (of dimension $ncomp \times ncomp$). For differential-algebraic systems $\underline{\underline{M}}$ is singular. When each component has an associated rate equation then $\underline{\underline{M}}$ could be written as an identity matrix $\underline{I}$. Frequently when the number of components $ncomp$ is more than the number of reactions $nreact$, all the ODEs are not independent. In other words, the rank of the stoichiometric matrix is less than $ncomp$. In such cases it is sometimes computationally advantageous to replace the dependent ODEs with algebraic conservation relations. These relations and the corresponding matrix $\underline{\underline{M}}$ are derived by a Gauss-Jordan elimination procedure on the stoichiometric matrix augmented by the identity matrix. If $\underline{R}_R$ is the vector (of length equal to $nreact$) of reaction rates then, the stoichiometric matrix $\underline{\underline{\nu}}$ (of dimensions $ncomp \times nreact$) is defined by,

$$\underline{I} \frac{d\underline{Y}}{dt} = \underline{\underline{\nu}} \underline{R}_R. \quad (3.2)$$

A Gauss-Jordan elimination process on $\underline{\underline{\nu}}$ and the same row transformations on $\underline{I}$ are carried out. The Gauss-Jordan elimination process is continued until no more pivots can be found in $\underline{\underline{\nu}}^*$. The result is,

$$\underline{\underline{M}} \frac{d\underline{Y}}{dt} = \underline{\underline{\nu}}^* \underline{R}_R \equiv \underline{F}(\frac{d\underline{Y}}{dt}, \underline{Y}, t). \quad (3.3)$$

All the ODEs in the above equation with a zero right-hand side correspond to invariants or conservation laws. Those ODEs can be replaced by the corresponding conservation laws. The number of such conservation laws is equal to,

$$ncomp - \text{rank of } \underline{\nu}$$

$\underline{M}$ is nonsingular when all the reactions are considered kinetic and used as such. If some of them are replaced by conservation laws, $\underline{M}$ becomes singular. The DAE system obtained by replacing the dependent ODEs with algebraic conservation laws is well behaved and is always of unit index [15].

Equilibrium reactions When a reaction is so fast (relative to the other temporal processes in the model) that it can be considered instantaneous then the fast reactions may be replaced by their equilibrium relations. Proper formulation of the system of equations to solve is necessary and important in this case. This was originally pointed out for a more general case by Sorensen and Stewart [30]. White and Seider [21] later used this formulation technique for the combustion of hexane. Here we use a modified form of this scheme so that it yields a consistent DAE system. We start as before with Equation 3.2 and proceed to Gauss-Jordan elimination. But pivots are first found for those columns in $\underline{\nu}$ that correspond to fast/equilibrium reactions. This guarantees that each of those fast reaction rates will appear in only one equation (which can be replaced by the equilibrium relation). After all the columns associated with the fast/equilibrium reactions are first considered for pivots, Gauss-Jordan process may be completed until no more pivots can be found. The resulting system of equations consists of,

- a N_e ODEs (= number of equilibrium/fast reactions) with a possibly indeterminate right-hand side (since their rates may be infinitely large),
- b N_k ODEs (= rank of $\underline{\nu}^* - N_e$) and,
- c N_c algebraic conservation laws (= $ncomp - N_e - N_k$).

Now if we decide to treat any of the fast reactions as being in equilibrium, we replace those ODEs that contain fast reaction rates by the equilibrium relations. We will then have a differential-algebraic system. The index of the resulting system can be deduced by a Singular Value Decomposition of $\underline{M}$ to convert it to an all ODE system [3, 15]. The index of the reaction system can change with time as the simulation progresses and it may be different at different parts of the domain. This version of RAFT assumes that the index of the reaction system is never greater than unity all through the simulation and at all spatial locations.

Example The above procedure for the formulation of the differential-algebraic reaction system is illustrated through an example. Consider the following arbitrary set of reactions, with corresponding reaction rates (written as per mass-action law). Here a is the concentration of A etc...

There are 4 reactions ($nreact = 4$) and 8 components ($ncomp = 8$). The rate of a reaction is interpreted as the rate of "consumption" of a unit amount of any "reactant" (left-hand side members). Let R_1, R_2, R_3 and R_4 be the rates associated with each reaction. Of these let us say that k_{f2}, k_{b2} and k_{f4}, k_{b4} are so large that the 2nd and 4th reactions above may be considered to be essentially at equilibrium. If they are at equilibrium then the following equilibrium relations apply.

$$k_{e2}bd^5 - ef^2 \equiv 0 \tag{3.5a}$$

$$k_{e4}eg^2 - h^3 \equiv 0 \tag{3.5b}$$

$k_{e2}(\equiv \frac{k_{f2}}{k_{b2}})$ and $k_{e4}(\equiv \frac{k_{f4}}{k_{b4}})$ are the equilibrium constants for these two reactions. From the set of reactions in Equation 3.4 the following rates for individual components can be written.

$$\begin{aligned}
 \frac{da}{dt} &= -R_1 - R_3 \\
 \frac{db}{dt} &= -R_1 - R_2 \\
 \frac{dc}{dt} &= R_1 + R_3 \\
 \frac{dd}{dt} &= -5R_2 + R_3 \\
 \frac{de}{dt} &= R_2 - R_4 \\
 \frac{df}{dt} &= 2R_2 - R_3 \\
 \frac{dg}{dt} &= -2R_4 \\
 \frac{dh}{dt} &= -2R_3 + 3R_4
 \end{aligned} \tag{3.6}$$

The above system of equations can be written in the form of Equation 3.2.

$$\left\{ \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right\} \left\{ \begin{array}{c} \frac{da}{dt} \\ \frac{db}{dt} \\ \frac{dc}{dt} \\ \frac{dd}{dt} \\ \frac{de}{dt} \\ \frac{df}{dt} \\ \frac{dg}{dt} \\ \frac{dh}{dt} \end{array} \right\} = \left\{ \begin{array}{cccc} -1 & 0 & -1 & 0 \\ -1 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & -5 & 1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 2 & -1 & 0 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 3 \end{array} \right\} \left\{ \begin{array}{c} R_1 \\ R_2 \\ R_3 \\ R_4 \end{array} \right\} \quad (3.7)$$

We apply Gauss-Jordan elimination process to the matrix $\underline{v}$ on the right side of the above equation. Since R_2 and R_4 are large we first obtain pivots for the second and fourth columns of $\underline{v}$. We also apply identical row transformations on $\underline{I}$ on the left side of the above equation. Obtaining a pivot in the second column of $\underline{v}$,

$$\left\{ \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -5 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right\} \left\{ \begin{array}{c} \frac{da}{dt} \\ \frac{db}{dt} \\ \frac{dc}{dt} \\ \frac{dd}{dt} \\ \frac{de}{dt} \\ \frac{df}{dt} \\ \frac{dg}{dt} \\ \frac{dh}{dt} \end{array} \right\} = \left\{ \begin{array}{cccc} -1 & 0 & -1 & 0 \\ -1 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 5 & 0 & 1 & 0 \\ -1 & 0 & 0 & -1 \\ -2 & 0 & -1 & 0 \\ 0 & 0 & 0 & -2 \\ 0 & 0 & -2 & 3 \end{array} \right\} \left\{ \begin{array}{c} R_1 \\ R_2 \\ R_3 \\ R_4 \end{array} \right\}. \quad (3.8)$$

Obtaining an another pivot in column 4 we get,

$$\left\{ \begin{array}{cccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -5 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & -2 & 0 & 0 & -2 & 0 & 1 & 0 \\ 0 & 3 & 0 & 0 & 3 & 0 & 0 & 1 \end{array} \right\} \left\{ \begin{array}{c} \frac{da}{dt} \\ \frac{db}{dt} \\ \frac{dc}{dt} \\ \frac{dd}{dt} \\ \frac{de}{dt} \\ \frac{df}{dt} \\ \frac{dg}{dt} \\ \frac{dh}{dt} \end{array} \right\} = \left\{ \begin{array}{cccc} -1 & 0 & -1 & 0 \\ -1 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 5 & 0 & 1 & 0 \\ -1 & 0 & 0 & -1 \\ -2 & 0 & -1 & 0 \\ 2 & 0 & 0 & 0 \\ -3 & 0 & -2 & 0 \end{array} \right\} \left\{ \begin{array}{c} R_1 \\ R_2 \\ R_3 \\ R_4 \end{array} \right\}. \quad (3.9)$$

Continuing to find all possible pivots; substituting for R_1, R_2, R_3 and R_4 ; and multiplying out the right-hand side we get,

$$\frac{1}{4} \begin{pmatrix} -1 & 5 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 20 & -20 & 0 & 4 & 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 1 & 4 & 0 & 0 & 0 \\ -3 & 3 & 0 & 1 & 0 & 4 & 0 & 0 \\ -2 & 2 & 0 & -2 & -8 & 0 & 4 & 0 \\ -7 & 7 & 0 & 1 & 12 & 0 & 0 & 4 \end{pmatrix} \begin{pmatrix} \frac{da}{dt} \\ \frac{db}{dt} \\ \frac{dc}{dt} \\ \frac{dd}{dt} \\ \frac{de}{dt} \\ \frac{df}{dt} \\ \frac{dg}{dt} \\ \frac{dh}{dt} \end{pmatrix} = \begin{pmatrix} -(k_{f1}ab - k_{b1}c) \\ -(k_{f2}bd^5 - k_{b2}ef^2) \\ 0 \\ -4(k_{f3}afh^2 - k_{b3}cd) \\ -(k_{f4}eg^2 - k_{b4}h^3) \\ 0 \\ 0 \\ 0 \end{pmatrix}. \quad (3.10)$$

The matrix $\underline{\nu}^*$ in the above equation has a rank of 4. So there are 4 independent ODEs. Of these, two ODEs corresponding to rows 2 and 5 contain very high rates and can be replaced by equilibrium relations ($N_e = 2$). The ODEs corresponding to rows 3, 6, 7, and 8 can be replaced by conservation laws ($N_c = 4$).

If all reactions are modeled as kinetic then any one of equations 3.7 through 3.10 can be used.

If it is desired to replace the dependent ODEs (corresponding to rows 3, 6, 7, and 8) by conservation laws then the new equations in their place are,

$$4a + 4c = 4a_0 + 4c_0, \quad (3.11a)$$

$$-3a + 3b + d + 4f = -3a_0 + 3b_0 + d_0 + 4f_0, \quad (3.11b)$$

$$-2a + 2b - 2d - 8e + 4g = -2a_0 + 2b_0 - 2d_0 - 8e_0 + 4g_0, \quad (3.11c)$$

$$-7a + 7b + d + 12e + 4h = -7a_0 + 7b_0 + d_0 + 12e_0 + 4h_0, \quad (3.11d)$$

where $a_0, b_0, c_0, d_0, e_0, f_0$, and g_0 are the initial (beginning of a time-step) values. So the complete system of equations to be solved in this case is the following.

$$\begin{pmatrix} -1/4 & 5/4 & 0 & -1/4 & 0 & 0 & 0 & 0 \\ 1/4 & -1/4 & 0 & 1/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & -5 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1/4 & -1/4 & 0 & 1/4 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{da}{dt} \\ \frac{db}{dt} \\ \frac{dc}{dt} \\ \frac{dd}{dt} \\ \frac{de}{dt} \\ \frac{df}{dt} \\ \frac{dg}{dt} \\ \frac{dh}{dt} \end{pmatrix} =$$

$$\left\{ \begin{array}{l} -k_{f1}ab - k_{b1} \\ -(k_{f2}bd^5 - k_{b2}ef^2) \\ (4a + 4c) - (4a_0 + 4c_0) \\ -4(k_{f3}afh^2 - k_{b3}cd) \\ -(k_{f4}eg^2 - k_{b4}h^3) \\ -3a + 3b + d + 4f - (-3a_0 + 3b_0 + d_0 + 4f_0) \\ -2a + 2b - 2d - 8e + 4g - (-2a_0 + 2b_0 - 2d_0 - 8e_0 + 4g_0) \\ -7a + 7b + d + 12e + 4h - (-7a_0 + 7b_0 + d_0 + 12e_0 + 4h_0) \end{array} \right\} \cdot \quad (3.12)$$

When it is also desired to approximate the fast reactions 2 and 4 as in equilibrium then the appropriate system of equations is formulated by replacing those ODEs that contain R_2 or R_4 in their right-hand side, with the equilibrium relations 3.5a and 3.5b respectively. There are two such ODEs (one containing R_2 and the other R_4); equations 2 and 5 in the above system of equations. The strategy for pivot selection was such that each of the fast reaction rates appears in only **one** equation. This guarantees that we can replace (if we choose to) all such equations with their corresponding equilibrium relations. This results in the following system of equations.

$$\left\{ \begin{array}{cccccccc} -1/4 & 5/4 & 0 & -1/4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & -5 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right\} \left\{ \begin{array}{l} \frac{da}{dt} \\ \frac{db}{dt} \\ \frac{dc}{dt} \\ \frac{dd}{dt} \\ \frac{de}{dt} \\ \frac{df}{dt} \\ \frac{dg}{dt} \\ \frac{dh}{dt} \end{array} \right\} = \left\{ \begin{array}{l} -(k_{f1}ab - k_{b1}c) \\ k_{e2}bd^5 - ef^2 \\ (4a + 4c) - (4a_0 + 4c_0) \\ -4(k_{f3}afh^2 - k_{b3}cd) \\ k_{e4}eg^2 - h^3 \\ -3a + 3b + d + 4f - (-3a_0 + 3b_0 + d_0 + 4f_0) \\ -2a + 2b - 2d - 8e + 4g - (-2a_0 + 2b_0 - 2d_0 - 8e_0 + 4g_0) \\ -7a + 7b + d + 12e + 4h - (-7a_0 + 7b_0 + d_0 + 12e_0 + 4h_0) \end{array} \right\} \cdot \quad (3.13)$$

Index of the reaction system Whether the index of the above reaction system is unity or greater can be deduced by converting the DAE system (for example Equation 3.10) to an ODE system. Using a singular value decomposition of $\underline{M}$, Equation 3.10

can be written as,

$$\frac{d\underline{P}}{dt} = \underline{S}(\underline{P}, \underline{Q}), \quad (3.14a)$$

$$0 = \underline{W}(\underline{P}, \underline{Q}), \quad (3.14b)$$

with appropriately defined $\underline{P}$ and $\underline{Q}$. $d\underline{Q}/dt$ is then obtained by,

$$\frac{d\underline{Q}}{dt} = -\underline{W}_Q^{-1} \underline{W}_P \frac{d\underline{P}}{dt}. \quad (3.14c)$$

$\underline{W}_Q$ is the jacobian $\frac{d\underline{W}}{d\underline{Q}}$ and $\underline{S}_P$ is the jacobian $\frac{d\underline{S}}{d\underline{P}}$. If $\underline{W}_Q$ has a bounded inverse in a neighborhood of the solution then the DAE system is of unit index.

Using the singular value decomposition theorem in the above example (Equation 3.10),

$$\underline{M} = \underline{U} \underline{D} \underline{V}^T, \quad (3.14d)$$

where,

$$\underline{U} = \begin{Bmatrix} -0.1376 & -0.0436 & -0.3932 & 0.0087 & 0.0712 & -0.9052 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.9955 & 0.0945 \\ 0.0903 & 0.0201 & -0.9157 & -0.0878 & -0.0294 & 0.3800 & 0 & 0 \\ 0.9058 & 0.2490 & 0.0335 & -0.0769 & -0.2752 & -0.1867 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -0.0945 & 0.9955 \\ -0.1325 & -0.0169 & 0.0720 & -0.9767 & -0.1483 & -0.0314 & 0 & 0 \\ -0.0660 & -0.5969 & -0.0141 & 0.1383 & -0.7873 & -0.0156 & 0 & 0 \\ -0.3614 & 0.7610 & -0.0188 & 0.1148 & -0.5259 & -0.0138 & 0 & 0 \end{Bmatrix},$$

$$\underline{D} = \begin{Bmatrix} 7.8115 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.7042 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.3134 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0656 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9620 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5341 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.0 \end{Bmatrix}, \text{ and}$$

$$\underline{V} = \begin{Bmatrix} 0.6936 & 0.0690 & -0.5055 & -0.0112 & 0.0023 & -0.5084 & 0 & 0 \\ -0.6997 & -0.0753 & -0.4911 & -0.0630 & 0.0411 & -0.4749 & -0.0859 & -0.1579 \\ 0.0116 & 0.0054 & -0.6972 & -0.0824 & -0.0306 & 0.7114 & 0 & 0 \\ 0.1088 & 0.2010 & 0.1158 & -0.3413 & -0.0705 & 0.0677 & -0.4297 & -0.7897 \\ -0.1219 & 0.9386 & -0.0215 & 0.0636 & -0.0032 & -0.0190 & 0.3146 & 0.0189 \\ -0.017 & -0.0046 & 0.0548 & -0.9166 & -0.1541 & -0.0588 & 0.1719 & 0.3159 \\ -0.0085 & -0.1611 & -0.0107 & 0.1297 & -0.8184 & -0.0293 & 0.4572 & -0.2781 \\ -0.0463 & 0.2054 & -0.0143 & 0.1077 & -0.5467 & -0.0258 & -0.6858 & 0.4171 \end{Bmatrix}.$$

$\underline{\underline{D}}$ can be further decomposed as

$$\underline{\underline{D}} = \underline{\underline{D}}^* \begin{Bmatrix} \underline{\underline{I}} & 0 \\ 0 & 0 \end{Bmatrix}, \quad (3.14e)$$

where,

$$\underline{\underline{D}}^* = \begin{Bmatrix} 7.8115 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3.7042 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.3134 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.0656 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9620 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.5341 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \end{Bmatrix}, \quad (3.14f)$$

and $\underline{\underline{I}}$ is a $(N_c - N_e) \times (N_c - N_e) = (6 \times 6)$ sized identity matrix. Using $\underline{\underline{V}}^T \underline{\underline{Y}} \equiv \{\underline{\underline{P}}, \underline{\underline{Q}}\}$ (where $\underline{\underline{P}}$ is of length $N_c - N_e = 6$, and $\underline{\underline{Q}}$ is of length $N_e = 2$), Equation 3.14d and, Equation 3.14e in Equation 3.10 (after replacing the 2nd and 5th rows with equilibrium relations) we get,

$$\begin{Bmatrix} \underline{\underline{I}} & 0 \\ 0 & 0 \end{Bmatrix} \begin{Bmatrix} d\underline{\underline{P}}/dt \\ d\underline{\underline{Q}}/dt \end{Bmatrix} = \underline{\underline{D}}^{*-1} \underline{\underline{U}}^{-1} \begin{Bmatrix} -(k_{f1}ab - k_{b1}c) \\ k_{e2}bd^5 - ef^2 \\ 0 \\ -4(k_{f3}afh^2 - k_{b3}cd) \\ k_{e4}eg^2 - h^3 \\ 0 \\ 0 \\ 0 \end{Bmatrix} = \begin{Bmatrix} 0.0176(k_{f1}ab - k_{b1}c) + 0.4638(k_{b3}cd - k_{f3}afh^2) \\ 0.0118(k_{f1}ab - k_{b1}c) + 0.2689(k_{b3}cd - k_{f3}afh^2) \\ 0.2994(k_{f1}ab - k_{b1}c) + 0.1021(k_{b3}cd - k_{f3}afh^2) \\ 0.0082(k_{b1}c - k_{f1}ab) + 0.2887(k_{f3}afh^2 - k_{b3}cd) \\ 0.074(k_{b1}c - k_{f1}ab) + 1.1441(k_{f3}afh^2 - k_{b3}cd) \\ 1.6947(k_{f1}ab - k_{b1}c) + 1.3979(k_{f3}afh^2 - k_{b3}cd) \\ 0.9955(k_{e1}bd^5 - ef^2) + 0.0945(h^3 - k_{e2}eg^2) \\ 0.0945(k_{e1}bd^5 - ef^2) + 0.9955(k_{e2}eg^2 - h^3) \end{Bmatrix} = \begin{Bmatrix} \underline{\underline{S}} \\ \underline{\underline{W}} \end{Bmatrix}$$

The jacobian $\underline{\underline{W}}_Q$ (2×2 matrix) is,

$$W_Q(1,1) = -0.0856k_{e1}d^5 - 2.1391k_{e1}bd^4 - 0.3132f^2 - 0.0297k_{e2}g^2 -$$

$$\begin{aligned}
& 0.3423ef - 0.0864k_{eq2}eg - 0.1945h^2 \\
W_Q(1,2) = & -0.1572k_{eq1}d^5 - 3.9309k_{eq1}bd^4 - 0.0188f^2 - 0.0018k_{eq2}g^2 - \\
& 0.6289ef + 0.0526k_{eq2}eg + 0.1183h^2 \\
W_Q(2,1) = & -0.0081k_{eq1}d^5 - 0.2031k_{eq1}bd^4 - 0.0297f^2 + 0.3132k_{eq2}g^2 - \\
& 0.0325ef + 0.9104k_{eq2}eg + 2.0483h^2 \\
W_Q(2,2) = & -0.0149k_{eq1}d^5 - 0.3732k_{eq1}bd^4 - 0.0018f^2 + 0.0188k_{eq2}g^2 - \\
& 0.0597ef - 0.5536k_{eq2}eg - 1.2457h^2
\end{aligned}$$

3.4.2 Solution of initial-value DAE systems

Given the initial values (concentrations at the beginning of the time-step), Equation 3.1 is solved for concentrations at the end of the time-step. If matrix $\underline{M}$ is singular (when some of the ODEs are replaced by conservation laws or equilibrium relations) then the initial concentrations are **required** to satisfy the algebraic constraints. The initial condition is a solution of the system of equations and as such should satisfy the equilibrium relations. Otherwise the DAE solvers used in RAFT will mostly fail. When the only algebraic relations are due to the conservation laws then this will hold for any set of initial values. But when equilibrium relations are present, care should be taken to provide good initial values (equilibrated concentrations). If Equation 3.13 is the reaction system to be solved then the initial values b_0, d_0, e_0, f_0, g_0 and h_0 must satisfy $k_{e2}b_0d_0^5 - e_0f_0^2 = 0$ and $k_{e4}e_0g_0^2 - h_0^3 = 0$.

Five different solvers are provided for solving the initial value ODE, DAE systems. They are the following.

RK4	A fourth order Runge-Kutta method. This can only be used when all the reactions are kinetic. $\underline{M} \equiv \underline{I}$. This is satisfactory for non-stiff ODE systems.
LSODA	This is part of ODEPACK. It can automatically detect stiffness and choose an appropriate method. This can only be used when all the reactions are kinetic [18]. $\underline{M} \equiv \underline{I}$.
RADAU5	This can solve the general DAE system as in Equation 3.1. It employs implicit Runge-Kutta methods [15].
DASSL	This can solve the general DAE system as in Equation 3.1. This uses backward differentiation formulas [29, 8].
LSODI	This is part of ODEPACK. This too can solve the general DAE system as in Equation 3.1 [18].

To reduce the complexity of the overall code, RAFT uses default values for most of the options in each of the methods LSODA, RADAU5, DASSL, and LSODI. With some modifications to RAFT that would allow user specified options for each method, it is possible to improve their performance for some problems. Original references to these packages should be consulted, if problems arise in their use and RAFT crashes.

Equilibration

The preceding section stresses that the initial condition for a DAE system should satisfy the algebraic constraints for the DAE solvers to succeed. This is not a concern when a batch system is being modeled. Once an initial concentration vector is chosen to obey the algebraic constraints, it will continue to obey those constraints, since the concentration at the beginning of any succeeding time-step is the solution of the DAE system from the previous time-step. This is not the case however when you have flow and transport in addition. Since we use operator splitting, the initial concentration vector for the DAE system is the concentration at the end of transport, which will not in general obey the algebraic constraints. So the use of DAE solvers with this initial concentration vector will fail. The initial concentration vector should be "equilibrated" before being used as an initial condition for the DAE solvers.

Equilibration is essentially a process of redistribution of the species that take part in equilibrium reactions. The problem can be posed as the solution of a system of nonlinear equations. This system of nonlinear equations can be derived according to the scheme described in Section 3.4.1, by considering **only** the equilibrium reactions. The previous example is considered here for illustration.

Example The reaction system to consider is the 2nd and 4th reactions of the system 3.4.

We have $n_{react} = 2$ and $n_{comp} = 6$. The individual reaction rates can be written as the following.

$$\begin{aligned} \frac{db}{dt} &= -R_2 \\ \frac{dd}{dt} &= -5R_2 \\ \frac{de}{dt} &= R_2 - R_4 \end{aligned} \quad (3.16)$$

$$\begin{aligned}\frac{df}{dt} &= 2R_2 \\ \frac{dg}{dt} &= -2R_4 \\ \frac{dh}{dt} &= 3R_4\end{aligned}$$

The above system of equations can be written in the form of Equation 3.2.

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{db}{dt} \\ \frac{dd}{dt} \\ \frac{de}{dt} \\ \frac{df}{dt} \\ \frac{dg}{dt} \\ \frac{dh}{dt} \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ -5 & 0 \\ 1 & -1 \\ 2 & 0 \\ 0 & -2 \\ 0 & 3 \end{pmatrix} \begin{Bmatrix} R_2 \\ R_4 \end{Bmatrix} \quad (3.17)$$

We apply Gauss-Jordan elimination process as before to the matrix $\underline{v}$ and effect identical row transformations on the identity matrix $\underline{I}$. The result is,

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -5 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 0 \\ -2 & 0 & -2 & 0 & 1 & 0 \\ 3 & 0 & 3 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{db}{dt} \\ \frac{dd}{dt} \\ \frac{de}{dt} \\ \frac{df}{dt} \\ \frac{dg}{dt} \\ \frac{dh}{dt} \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & 0 \\ 0 & -1 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{Bmatrix} R_2 \\ R_4 \end{Bmatrix} = \begin{pmatrix} -R_2 \\ 0 \\ -R_4 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (3.18)$$

Equations 2,4,5 and 6 have a zero right-hand side in the above equation. So they represent conservation laws. They can be replaced by the following algebraic relations.

$$-5b + d = -5b_0 + d_0, \quad (3.19a)$$

$$2b + f = 2b_0 + f_0, \quad (3.19b)$$

$$-2b - 2e + g = -2b_0 - 2e_0 + g_0, \quad (3.19c)$$

$$3b + 3e + h = 3b_0 + 3e_0 + h_0 \quad (3.19d)$$

where b_0, d_0, e_0, f_0 , and g_0 are the values just after transport and before equilibration. Equations 1 and 3 contain infinitely large reaction rate expressions and so are replaced by equilibrium relations. So the original ODE system, Equation 3.17 is reduced to a pure algebraic problem. The final set of equations to solve are,

$$k_{e2}bd^5 - ef^2 = 0,$$

$$-5b + d - (-5b_0 + d_0) = 0,$$

$$\begin{aligned}
k_{e4}eg^2 - h^3 &= 0, \\
2b + f - (2b_0 + f_0) &= 0, \\
-2b - 2e + g - (-2b_0 - 2e_0 + g_0) &= 0, \\
3b + 3e + h - (3b_0 + 3e_0 + h_0) &= 0.
\end{aligned}
\tag{3.20}$$

It is the equilibrium relations that can make the system nonlinear (as in the above example). The conservation relations are always linear. Solution of the above system yields an equilibrated set of concentrations $\{b,d,e,f,g,h\}$ which together with the transported concentrations $\{a,c\}$ form a consistent initial condition the DAE system, Equation 3.13.

RAFT allows for two methods for the solution of the nonlinear equations (for example Equation 3.20 above). These two are NNES [4] and INTBIS [22]. Additional techniques may be added in later versions of RAFT. NNES implements a variety of techniques based on newton methods and minimization. INTBIS requires that all equations be representable as polynomials, with the unknowns raised to **integer** powers. INTBIS is based on a generalized bisection approach to find the roots. NNES will find the nearest root to the initial guess whereas INTBIS can find all the real roots in a given search region. If INTBIS finds multiple roots which are feasible (nonnegative, physical etc...) then RAFT takes that root which is closest to the transported concentration. Such a situation is however not very common. To reduce the complexity of the overall code, RAFT uses default values for most of the options in each method. With some modifications to RAFT that would allow user specified options for NNES and INTBIS, it is possible to improve their performance for some problems.

3.4.3 Preparation of the file `rxns.f`

The file `rxns.f` defines the reaction system (Equation 3.1). It consists of the following subroutines.

`physrxns`

If the "constants" that appear in reaction rate expressions or equilibrium relations change with time then this routine can be used to specify how they change. If reactions are present, this routine will be called after every time-step, even if the "constants" do not change with time. So if the "constants" do not change with time, the body of this subroutine should be empty.

`inityprime`

When the reaction system is differential-algebraic as in Equation 3.13, some of the solvers employed (DASSL, LSODI) need the initial values of derivatives which together with the initial conditions, satisfy Equation 3.1. An estimate for the derivatives can be made from Equations

tion 3.9 by inverting the matrix on the left-hand side (this can always be inverted since it is a product of elementary matrices) and taking the net equilibrium reaction rates to be zero (in the example this means $R_2 = R_3 \equiv 0$). A more accurate estimate can be obtained by converting the system to an index-0 (all ODE) system. In the example problem, Equations 3.14a and 3.14c can be used to estimate the initial rates of reaction.

```
subroutine inityprime (t,y,initydot,ncomp,rxnk,
nrxnconsts,nalgebraic,conc0,porosity)
```

Here t refers to time, y is a vector (of length $ncomp$) of concentrations, $rxnk$ is an array (of length $nrxnconsts$) containing the reaction constants and $initydot$ is the vector (of length $ncomp$) dy/dt . $nalgebraic$ is the number of algebraic relations in the reaction system and $conc0$ is the vector of initial concentrations.

setmatrixM

This routine defines the matrix $\underline{M}$ of Equation 3.1.

```
subroutine setmatrixM (matrixM,ncomp)
```

$matrixM$ has the dimensions of $(ncomp \times ncomp)$.

defrxns

This routine defines the right-hand side of Equation 3.1.

```
subroutine defrxns (y,ncomp,nrxnconsts,rxnk,func,
porosity,conc0)
```

Here y is vector of concentrations, $func$ is the right-hand side vector that this routine provides and $conc0$ is the vector of initial concentrations.

jacobian

All the solvers use a jacobian when the system is stiff (except RK4 which cannot solve stiff systems). Jacobian J is defined by,

$$J = \frac{dF}{dY} \quad (J_{ij} = \frac{\partial F_i}{\partial Y_j}). \quad (3.21)$$

It is best to use an analytic form for J if it is available. In this case this subroutine is used to supply J . When an analytic form is not available, all the solvers can compute a numerical approximation to J .

```
subroutine jacobian (ncomp,t,y,pd,rxnk,nrxnconsts)
```

Here *pd* is the jacobian matrix (of dimensions *ncomp* × *ncomp*).

The entire process i.e., the generation of the reaction system as in Equation 3.13 (from Equations 3.4 and the equilibrium relations Equation 3.5a and 3.5b) and the generation of the code for the routines `setmatrixM`, `defrxns`, `inityprime`, and `jacobian` could be automated with a symbolic computational language like MAPLE. Input to this MAPLE program is the reaction system (like Equation 3.4), which reactions are to be treated as equilibrium and finally the rate expressions so that the jacobian *J* can be generated. Appendix A gives the MAPLE procedure `rxninput` for this example.

3.4.4 Preparation of the file `preeqrxns.f`

As described in Section 3.4.2, when transport is present, the transported solution needs to be equilibrated before it can be used as an initial condition to the DAE system. The file `preeqrxns.f` defines the nonlinear system to be solved for roots. It consists of the following subroutines.

`nnesfcn`

This subroutine defines the nonlinear equations to be solved by NNES.

```
subroutine nnefcn (overfl,n,func,y)
```

n is the number of unknowns. *y* is the vector of unknowns. This routine calculates the values of the *n* functions at this *y*, and places them in the vector *func*. When *func* is zero for some *y**, then *y** is the solution of the system.

`nnesjacob`

This subroutine defines the jacobian of the nonlinear system of equations to be solved by NNES. If *F* is the vector of nonlinear equations and *y* is the vector of unknowns then the jacobian *J* is defined by,

$$J = \frac{dF}{dy} \quad (J_{ij} = \frac{\partial F_i}{\partial y_j}).$$

```
subroutine nnesjacob (overfl,n,pd,y)
```

n is the number of unknowns, *y* is the vector (of length *n*) of unknown values at which the jacobian *pd* (of dimension *n* × *n*) will be evaluated.

intdefinerns

This subroutine defines the nonlinear system for INTBIS. INTBIS expects each equation to be a polynomial with the unknowns raised to integer powers.

```
subroutine intdefinerns (dummaxncomp, NUMT,  
KDEG1, A1, neqpreeq)
```

dummaxncomp is equal to *maxncomp* defined in **sizes.h**. *numt* is a vector that contains the "number of terms" in each nonlinear equation. *kdeg1* is a three-dimensional array that supplies the (integer) degree of each unknown in each term for each equation. *a1* is two-dimensional array that has the coefficients multiplying each term in each equation. *numt*, *kdeg1*, and *a1* are defined in this subroutine. Together they can define any polynomial system. User is referred to [22] and the package INTBIS for details. *neqpreeq* is the number of equations in the equilibration system.

Again the generation of the reaction system as in Equation 3.20 and the generation of the code for the routines **nnesfcn**, **nnesjacob**, and **intdefinerns** could be automated with a symbolic computational language like MAPLE. The input file **rxninput** to the MAPLE program is given in Appendix A.

3.4.5 Description of the file **datarn**

The file **datarn** chooses the solver to use and supplies the various constants in *F*.

Card # 1: 1 entry.

odeflag An integer. This is a flag that identifies the solver to use. It can take the following values.

odeflag = 0 Use RK4.

odeflag = 1 Use LSODA.

odeflag = 2 Use RADAU5.

odeflag = 3 Use DASSL.

odeflag = 4 Use LSODI.

When *M* is singular then *odeflag* has to be > 1.

Card # 2: 3 entries.

nalgebraic An integer. This is the number of algebraic relations in the reaction system (Equation 3.1): This should be > 0 when there are some conservation laws or equilibrium relations. If zero then all the equations are ODEs.

neqpreeq An integer. This is number of nonlinear equations to be solved for equilibration.

preeqflag An integer flag. Can take values 1 or 2. This selects the solver to use for equilibration. When equal to 1, NNES is used and when equal to 2, INTBIS is used.

Card # 3: 3 entries.

eqcompId *neqpreeq* integers. These are the component "numbers" (identifications) that participate in equilibration.

Card # 4: 3 entries.

nnesftol Real. This is the variable "FTOL" of the package NNES. If each of the residuals in the nonlinear system are less than *nnesftol* then NNES assumes convergence has been achieved.

nnesmarit Integer. This is the variable "MAXIT" of the package NNES. This is the maximum number of iterations allowed to achieve convergence.

nnesoutput Integer. This is the variable "OUTPUT" of the package NNES. If *nnesoutput* is not equal to zero then some information from NNES will be printed to the file nnesout.

Card # 5: 4 entries.

intmaxft Integer. This is the variable "MAXFT" of the package INTBIS. Large values for *intmaxft* have greater chance of solving the system.

intprtcon Integer. This is the variable "PRTCON" of the package INTBIS. If *intprtcon* is not equal to zero the some information from INTBIS will be printed to the file bisecout.

intEPS Real. This is the variable "EPS" of the package INTBIS. This is related to the size of the smallest box to be produced by bisection.

intEPSF Real. This is the variable "EPSF" of the package INTBIS. This is the range tolerance.

Card # 6: Reals. The search region (box) for the roots of the equilibration system is given here. There are *negpreeq* lines with the *i*th line supplying the lower and upper bound for the *i*th unknown of the equilibration system.

Card # 7: *rtol*, *atol*

Real. Two entries per line and there are *ncomp* lines. *rtol(i)* gives the relative tolerance required for component *i*. *atol(i)* gives the absolute tolerance required for component *i*. The local error is reduced such that,

$$\text{abs}(\text{local error}) < \text{rtol}(i) \text{abs}(y(i)) + \text{atol}(i)$$

These tolerance limits are used by the solvers 1 through 4.

Card # 8: 1 entry.

RKsteps An integer. If *odeflag* = 0 then RK4 is used to integrate the ODE system. Each transport time-step Δt is divided into *RKsteps* sub time-steps to be used in this integration. More *RKsteps* means an accurate integration but also more CPU time. For a given set of reactions and a flow problem, the number of *RKsteps* to use should be determined by trial and error until a satisfactory convergent solution is obtained.

Card # 9: 2 entries. This applies if *odeflag* = 1.

lsodajt An integer flag. If equal to zero then the jacobian need not be supplied. If equal to unity then the jacobian needs to be supplied (in subroutine jacobian, in file `rxns.f`). When *lsodajt* is 0, the variable "jt" of LSODA is set to 2 and when *lsodajt* is 1, "jt" is set to 1, before calling LSODA.

- Integer. Specifies the maximum number of iterations allowed. This is the variable "iwork(6)" of LSODA.

Card # 10: 4 entries. This applies if *odeflag* = 2.

radaijac An integer flag. This is the variable "IJAC" of RADAU5. If equal to zero then the jacobian need not be supplied. If equal to unity then the jacobian needs to be supplied (in subroutine `jacobian`, in file `rxns.f`).

- An integer. This is the variable "IWORK(3)" of RADAU5. Specifies the maximum number of Newton iterations allowed.

- Real. This is the variable "WORK(3)" of RADAU5. If this quantity is small (≈ 0.001) then the jacobian is recomputed more often (this can help convergence). When jacobian evaluations are costly this quantity could be made big (≈ 0.1).
- Real. This is the variable "WORK(4)" of RADAU5. Specifies the stopping criterion for the Newton's method. Smaller values (≈ 0.01) can make the code slower but safer.

Card # 11: 3 entries. This applies if *odeflag* = 3.

dassljac An integer. A flag. This is the variable "INFO(5)" of DASSL. If equal to zero then the jacobian need not be supplied. If equal to unity then the jacobian needs to be supplied (in subroutine **jacobian**, in file **rxns.f**).

- An integer. This is the variable "INFO(10)" of DASSL. Specifies whether the nonnegativity of concentrations is to be imposed during the solution. If equal to zero then it will not be imposed and it will be imposed if equal to 1.
- An integer. This is the variable "INFO(11)" of DASSL. If equal to 0 then DASSL expects an estimate of initial derivative that satisfies Equation 3.1. If equal to 1 then an approximate estimate can be supplied.

Card # 12: 2 entries. This applies if *odeflag* = 4.

lsodimf An integer. A flag. If equal to zero then the jacobian need not be supplied. If equal to unity then the jacobian needs to be supplied (in subroutine **jacobian**, in file **rxns.f**). This flag is similar to the flag "mf" in LSODI. When *lsodimf* = 0, "mf" is set to 22 and when *lsodimf* = 1, "mf" is set to 21 before RAFT calls LSODI.

- An integer. Specifies the maximum number of iterations allowed. This is the variable "iwork(6)" of LSODI.

Card # 13: 1 entry.

nrnconsts An integer. This is the total number of constants¹ in the reactions.

Cards: As many as needed.

¹All constants appearing in the reaction rate expressions are taken to be real variables.

rxnk A real array of length *nrnxconsts*. The *nrnxconsts* constants appearing in reaction rate expressions have to be listed here. They will be denoted as *rxnk*(1) ... *rxnk*(*nrnxconsts*) in the code.

Example 1 Consider the reaction system Equation 3.13. The only subroutines that need to be supplied are `setmatrixM`, `defrxns`, `inityprime` and if possible `jacobian`. MAPLE was used to formulate the reaction problem, derive the final system of equations and also to generate the FORTRAN code to be included in the file `rxns.f`.

`setmatrixM` M has only six nonzero entries.

```

do 10 j = 1, ncomp
do 10 i = 1, ncomp
    matrixM(i,j) = 0.0
10 continue
matrixM(1,1) = -1/4
matrixM(1,2) = 5/4
matrixM(1,4) = -1/4
matrixM(4,1) = 5
matrixM(4,2) = -5
matrixM(4,4) = 1/4

```

`defrxns`

The right-hand side of Equation 3.13 is supplied here.

```

kf1 = rxnk(1)
kb1 = rxnk(2)
keq2 = rxnk(3)
kf3 = rxnk(4)
kb3 = rxnk(5)
keq4 = rxnk(6)
func(1) = -kf1*y(1)*y(2)+kb1*y(3)
func(2) = keq2*y(2)*y(4)**5-y(5)*y(6)**2
func(3) = y(1)-conc0(1)+y(3)-conc0(3)
func(4) = -4.D0*kf3*y(1)*y(6)*y(8)**2+4.D0*kb3*y(3)*y(4)
func(5) = keq4*y(5)*y(7)**2-y(8)**3
func(6) = -0.75D0*y(1)+0.75D0*conc0(1)+0.75D0*y(2)-
        0.75D0*conc0(2)+0.25D0*y(4)-0.25D0*conc0(4)+y(6)-
        conc0(6)
func(7) = -0.5D0*y(1)+0.5D0*conc0(1)+0.5D0*y(2)-0.5D0*

```

```

conc0(2)-0.5D0*y(4)+0.5D0*conc0(4)-2.D0*y(5)+2.D0*
conc0(5)+y(7)-conc0(7)
func(8) = -0.175D1*y(1)+0.175D1*conc0(1)+0.175D1*y(2)-
0.175D1*conc0(2)+0.25D0*y(4)-0.25D0*conc0(4)+
3.D0*y(5)-3.D0*conc0(5)+y(8)-conc0(8)

```

jacobian

Given the right-hand side $\underline{F}$ as above the jacobian J can be computed analytically from Equation 3.21.

$$pd(i,j) = \frac{\partial F_i}{\partial y_j} \quad (3.22)$$

```

kf1 = rxnk(1)
kb1 = rxnk(2)
ke2 = rxnk(3)
kf3 = rxnk(4)
kb3 = rxnk(5)
ke4 = rxnk(6)
do 10 j = 1, ncomp
do 10 i = 1, ncomp
    pd(i,j) = 0.0
10 continue
c
pd(1,1) = -kf1*y(2)
pd(1,2) = -kf1*y(1)
pd(1,3) = kb1
c
pd(2,2) = keq2*y(4)**5
pd(2,4) = 5*keq2*y(2)*y(4)**4
pd(2,5) = -y(6)**2
pd(2,6) = -2*y(5)*y(6)
c
pd(3,1) = 1
pd(3,3) = 1
c
pd(4,1) = -4.D0*kf3*y(6)*y(8)**2
pd(4,3) = 4.D0*kb3*y(4)
pd(4,4) = 4.D0*kb3*y(3)
pd(4,6) = -4.D0*kf3*y(1)*y(8)**2

```

```

pd(4,8) = -8.D0*kf3*y(1)*y(6)*y(8)
c
pd(5,5) = keq4*y(7)**2
pd(5,7) = 2*keq4*y(5)*y(7)
pd(5,8) = -3*y(8)**2
c
pd(6,1) = -3./4.
pd(6,2) = 3./4.
pd(6,4) = 1./4.
pd(6,6) = 1
c
pd(7,1) = -1./2.
pd(7,2) = 1./2.
pd(7,4) = -1./2.
pd(7,5) = -2
pd(7,7) = 1
c
pd(8,1) = -7./4.
pd(8,2) = 7./4.
pd(8,4) = 1./4.
pd(8,5) = 3
pd(8,8) = 1

```

inityprime

An estimate for initial values of derivatives is provided by from Equation 3.9 as explained earlier.

```

kf1 = rxnk(1)
kb1 = rxnk(2)
ke2 = rxnk(3)
kf3 = rxnk(4)
kb3 = rxnk(5)
ke4 = rxnk(6)
initydot(1) = -kf1*y(1)*y(2)+1.D0*kb1*y(3)-kf3*y(1)*y(6)*
y(8)**2+1.D0*kb3*y(3)*y(4)
initydot(2) = -kf1*y(1)*y(2)+0.1D1*kb1*y(3)+0.333D-20*
kf3*y(1)*y(6)*y(8)**2-0.333D-20*kb3*y(3)*y(4)
initydot(3) = 1.D0*kf1*y(1)*y(2)-kb1*y(3)+1.D0*kf3*y(1)*
y(6)*y(8)**2-kb3*y(3)*y(4)

```

```

initydot(4) = -0.2416667D-18*kf1*y(1)*y(2)+0.241667D-18*
             kb1*y(3)+0.1D1*kf3*y(1)*y(6)*y(8)**2-kb3*y(3)*y(4)
initydot(5) = 0.35D-19*kf1*y(1)*y(2)-0.35D-19*kb1*y(3)-
             0.1D-19*kf3*y(1)*y(6)*y(8)**2+0.1D-19*kb3*y(3)*y(4)
initydot(6) = 0.1D-18*kf1*y(1)*y(2)-0.1D-18*kb1*y(3)-
             kf3*y(1)*y(6)*y(8)**2+0.1D1*kb3*y(3)*y(4)
initydot(7) = -0.3D-19*kf1*y(1)*y(2)+0.3D-19*kb1*y(3)-
             0.2D-19*kf3*y(1)*y(6)*y(8)**2+0.2D-19*kb3*y(3)*y(4)
initydot(8) = -0.2D1*kf3*y(1)*y(6)*y(8)**2+0.2D1*kb3*y(3)*
             y(4)

```

Since $\underline{M}$ is singular we can only use the solvers RADAU5 (*odeflag* = 2) or DASSL (*odeflag* = 3) or LSODI (*odeflag* = 4).

nnesfcn

Equation 3.20 is the nonlinear system to be solved for equilibration.

```

ke2 = rxnk(3)
ke4 = rxnk(6)
do 10 i = 1, n
    inity(i) = conc0(eqcompId(i))
enddo
func(1) = keq2*y(1)*y(2)**5-y(3)*y(4)**2
func(2) = -5.D0*y(1)+5.D0*inity(1)+y(2)-inity(2)
func(3) = keq4*y(3)*y(5)**2-y(6)**3
func(4) = 2.D0*y(1)-2.D0*inity(1)+y(4)-inity(4)
func(5) = -2.D0*y(1)+2.D0*inity(1)-2.D0*y(3)+2.D0*inity(3)+
y(5)-inity(5)
func(6) = 3.D0*y(1)-3.D0*inity(1)+3.D0*y(3)-3.D0*inity(3)+
y(6)-inity(6)

```

nnesjacob

The jacobian corresponding to the above functions *fvec*(1) through *fvec*(6) and the variables {b,d,e,f,g,h} is computed by MAPLE as the following. Only the nonzero entries are given here.

```

ke2 = rxnk(3)
ke4 = rxnk(6)

do 10 j = 1, neqpreeq
do 10 i = 1, neqpreeq

```



```

        jac(i,j) = 0.0
10  continue

pd(1,1) = keq2*y(2)**5
pd(1,2) = 5*keq2*y(1)*y(2)**4
pd(1,3) = -y(4)**2
pd(1,4) = -2*y(3)*y(4)

pd(2,1) = -5.D0
pd(2,2) = 1

pd(3,3) = keq4*y(5)**2
pd(3,5) = 2*keq4*y(3)*y(5)
pd(3,6) = -3*y(6)**2

pd(4,1) = 2.D0
pd(4,4) = 1

pd(5,1) = -2.D0
pd(5,3) = -2.D0
pd(5,5) = 1

pd(6,1) = 3.D0
pd(6,3) = 3.D0
pd(6,6) = 1

```

intdefinernxns

Here we define the nonlinear system for INTBIS. numt, kdeg and a are defined. The first equation is $k_{e2}bd^5 - ef^2$. Since there are two terms in the equation numt(1) = 2. The degree of b (1st unknown) in the 1st term is 1, so kdeg(1,1,1) = 1. The degree of d (2nd unknown) in the 1st term is 5, so kdeg(1,2,1) = 5. It is easily checked that kdeg(1,3,2) = 1 and kdeg(1,4,2) = 2. Similarly a(1,1) = keq2 and a(1,2) = -1.0. The following code is generated by MAPLE. Only the nonzero values of kdeg and a are given here.

```

numt(1) = 2
numt(2) = 3
numt(3) = 2

```

```

numt(4) = 3
numt(5) = 4
numt(6) = 4

do 10 k = 1, 6
do 10 j = 1, 6
    a1(j,k) = 0.0
    do 20 i = 1, 6
        kdeg1(i,j,k) = 0
20 continue
10 continue

kdeg1(1,1,1) = 1
kdeg1(1,2,1) = 5
kdeg1(1,3,2) = 1
kdeg1(1,4,2) = 2

kdeg1(2,1,1) = 1
kdeg1(2,2,2) = 1

kdeg1(3,3,1) = 1
kdeg1(3,5,1) = 2
kdeg1(3,6,2) = 3

kdeg1(4,1,1) = 1
kdeg1(4,4,2) = 1

kdeg1(5,1,1) = 1
kdeg1(5,3,2) = 1
kdeg1(5,5,3) = 1

kdeg1(6,1,1) = 1
kdeg1(6,3,2) = 1
kdeg1(6,6,3) = 1

a1(1,1) = 0.1D1*keq2
a1(1,2) = -0.1D1

```

```

a1(2,1) = -0.5D1
a1(2,2) = 0.1D1
a1(2,3) = 5.D0*inity(1)-inity(2)

a1(3,1) = 0.1D1*keq4
a1(3,2) = -0.1D1

a1(4,1) = 0.2D1
a1(4,2) = 0.1D1
a1(4,3) = -2.D0*inity(1)-inity(4)

a1(5,1) = -0.2D1
a1(5,2) = -0.2D1
a1(5,3) = 0.1D1
a1(5,4) = 2.D0*inity(1)+2.D0*inity(3)-inity(5)

a1(6,1) = 0.3D1
a1(6,2) = 0.3D1
a1(6,3) = 0.1D1
a1(6,4) = -3.D0*inity(1)-3.D0*inity(3)-inity(6)

```

Example 2 Consider the following reactions that represent the aerobic biodegradation of dissolved hydrocarbons [6]. H stands for the hydrocarbon concentration, O for oxygen concentration and M for the concentration of microbes. Taking the rate expressions from [6],

$$\begin{aligned}
\frac{dH}{dt} &= -kM \left(\frac{H}{K_h + H} \right) \left(\frac{O}{K_o + O} \right) \\
\frac{dO}{dt} &= -kMP \left(\frac{H}{K_h + H} \right) \left(\frac{O}{K_o + O} \right) \\
\frac{dM}{dt} &= kMQ \left(\frac{H}{K_h + H} \right) \left(\frac{O}{K_o + O} \right) - bM
\end{aligned}$$

There are no equilibrium reactions. It is seen from above that $\frac{dH}{dt} - P\frac{dO}{dt} = 0$ and so one of those two ODEs may be replaced by the conservation law $(H - H_0 - P(O - O_0)) = 0$. However in this example all the equations are formulated as ODEs so that the use of $odeflag = 0$ or 1 may be demonstrated.

To incorporate these reactions into RAFT we note that $ncomp = 3$. We denote H as 1, O as 2, and M as 3. There are 6 constants, k, K_h, K_o, P, Q , and b . So $nrxnconsts = 6$.

We can assign, $rxnk(1) = k$, $rxnk(2) = K_h$, $rxnk(3) = K_o$, $rxnk(4) = P$, $rxnk(5) = Q$ and $rxnk(6) = b$. The subroutines needed are **setmatrixM**, **defrxns**, and if needed **jacobian**.

setmatrixM

$\underline{M} \equiv \underline{I}$.

```

do 10 j = 1, ncomp
do 10 i = 1, ncomp
    matrixM(i,j) = 0.0
10 continue
do 20 i = 1, ncomp
    matrixM(i,i) = 1.0
20 continue

```

defrxns

```

k = rxnk(1)
K_h = rxnk(2)
K_o = rxnk(3)
P = rxnk(4)
Q = rxnk(5)
b = rxnk(6)
H = y(1)
O = y(2)
M = y(3)
func(1) = -k*M*(H/(Kh+H))*(O/(Ko+O))
func(2) = -k*M*P*(H/(Kh+H))*(O/(Ko+O))
func(3) = k*M*Q*(H/(Kh+H))*(O/(Ko+O)) - b*M

```

jacobian

pd is a 3×3 matrix. Let,

$$\begin{aligned}
 x &\equiv \frac{\partial[\text{func}(1)]}{\partial H} = -\frac{kMO}{(Kh+H)(Ko+O)} + \frac{kMHO}{(Kh+H)^2(Ko+O)}, \\
 y &\equiv \frac{\partial[\text{func}(1)]}{\partial O} = -\frac{kM}{(Kh+H)(Ko+O)} + \frac{kMHO}{(Kh+H)(Ko+O)^2}, \\
 z &\equiv \frac{\partial[\text{func}(1)]}{\partial M} = \text{func}(1)/M.
 \end{aligned}$$

Now the jacobian J can be written as,

$$J = \begin{Bmatrix} x & y & z \\ Px & Py & Pz \\ -Qz & -Qy & -Qz - b \end{Bmatrix} \quad (3.23)$$

The above matrix can be turned into FORTRAN statements to make up the subroutine `jacobian`.

Knowing *nrnconsts*, and the numerical values of the constants, the file `datarxn` can be prepared. All five solvers can be used to solve this system. In the next chapter RK4 is used to solve this system in the context of a reactive flow simulation through a two-dimensional heterogeneous medium.

3.5 Parameter Estimation

In addition to the use of RAFT in computing the concentration distributions given the controlling parameters, RAFT can also be used to estimate the values of these parameters, given the experimental observations. The parameters are estimated by nonlinear least squares techniques. Three different solvers (all from Netlib) are provided for accomplishing nonlinear regression. They are LMDIF (part of MINPACK [27]), DQED [16], and NL2SOL [10]. When RAFT is used to estimate parameters, two input files, `lsqdata`, `lsqexptdata` and the subroutines `objfcn.f` and `computepredict.f` should be provided. These files are read only if *lsqfit* on Card # 1 is equal to 1. Among other things the file `lsqdata` specifies which parameters are to be estimated, their bounds and the method to use for regression. The file `lsqexptdata` provides the experimental data. The subroutine `computepredict.f` computes the predictions (to be compared against experimental observations) of the model from the concentration distributions generated by RAFT. Currently RAFT assumes that all experimental observations take place at a point in space and time. Observations involving multiple locations in space and time (for example the average mass flux across a section over a time period) can also be handled with some modifications to the code. The subroutine `objfcn.f` specifies the objective function, given the experimental observations and model predictions. When the regression is successful two output files `predictedc` and `fitparamlist` are generated. The file `predictedc` gives the final predictions by the model with the optimized parameters and the file `fitparamlist` documents the search through the parametric space and the associated reduction in the residuals. The description of each of these files follows.

`lsqdata`

Card # 1: 1 entry

lsqfitmethod Integer flag. Can take values from 0 through 2. Three different methods are available for use in nonlinear regression used to estimate parameters.

lsqfitmethod = 0 Use LMDIF.

lsqfitmethod = 1 Use DQED.

lsqfitmethod = 2 Use NL2SOL.

Card # 2: 1 entry

nfitparams Integer. The number of parameters to be estimated.

Card # 3: *nfitparams* entries.

fitparamID Integers. These are used to identify the parameters to fit. Currently RAFT allows the estimation of the following parameters whose identification numbers are indicated in the adjacent column.

<u>Identification Number</u>	<u>Parameter Description</u>
1	Longitudinal dispersivity, α_l .
2	Transverse dispersivity, α_t .
≥ 3 and $\leq 2+ntracetypes$	Retardation factors for trace-types 1 through <i>ntracetypes</i> .
$\geq 3+ntracetypes$ and $\leq 2+ntracetypes+nrxnconsts$	Reaction parameters 1 through <i>nrxnconsts</i> .

Card # 4: *nfitparams* lines with two entries per line. Each line gives the lower and upper bounds on the estimate for a parameter. The first line is for parameter *fitparamID*(1) and the last line is for the parameter *fitparamID*(*nfitparams*). These bounds are used by DQED and NL2SOL. LMDIF is written for unconstrained optimization. So RAFT assigns an arbitrarily large residual for LMDIF if the search for parameters crosses the boundaries.

Card # 5: 2 entries.

nob The total number of experimental observations. The experimental observations are typically the concentrations of components or some functions of the concentrations of components, at a point in space and time.

nfunc The number of such functions of concentrations. For example, if in a three component experiment, the measured data is the concentration of the first component and the sum of the concentrations of the other two components then *nfunc* = 2.

Card # 6: *nfunc* lines with two entries per line. Each line gives the range in 1 thru *nob* observations that correspond to the measurement of a function. For example if *nob* = 20 and *nfunc* is 2 as in the previous card then the first 12 observations could be the concentration of the first component and the following 8 observations could be the sum of the concentrations of 2nd and 3rd components. In this case the two lines for this card will be,
 1 12
 13 20

Card # 7: *epsfcn*. $\sqrt{\text{epsfcn}}$ is the fraction used in varying the parameters in the computation of jacobian. If *epsfcn* = 0.01 and a quantity *f* is a function of parameter *x* then,

$$\partial f / \partial x \approx (f(1.1x) - f(x)) / (0.1x).$$

Card # 8: When *lsqfitmethod* = 0 then LMDIF is used. The default values for these parameters may be good enough for most problems. The user may consult that routine to set the parameters on this card.

Card # 9: When *lsqfitmethod* = 1 then the program DQED is used. The first entry on this card gives the maximum number of iterations allowed and the second sets the flag for using linear or quadratic models. When this flag is equal to 1 then the use of quadratic model is suppressed.

Card # 10: When *lsqfitmethod* = 2 then the program NL2SOL is used. The first entry can take values 0 or 1. Only the evaluation of jacobian is different in these two cases. The second entry here is the maximum number of iterations allowed and the third entry controls printing by NL2SOL.

lsqexptdata

The experimental observations are provided in this file. These observations are usually functions of concentrations. Each observation consists of five quantities. The first three quantities give the spatial coordinates (x,y,z) of the point where the observation was taken; the fourth is the time at which the observation was taken and the fifth is the

value of the function/observation. The spatial location should be inside the domain or on the boundary and the observation time should be $\geq tstart$ and $\leq tfinal$.

comptecpredict.f

This is a subroutine that computes the observed quantities from the concentration distribution at the time of observation and at the spatial location of the observation. The experimental observations could be functions of concentrations of different components but usually at one spatial location and time. Experimental observations involving concentrations at different spatial locations (for example, mass) and different times can also be handled by some modifications to the code. **comptecpredict.f** is called as in the following.

```
subroutine comptecpredict (iloc,jloc,kloc,csol,funcindex,cpredict,  
obsindex,nxp,nyp,nzp,ncomp,nob)
```

Here *iloc,jloc,kloc* is the grid location of the observation, and *csol* contains the concentration distribution *csol(nxp,nyp,nzp,ncomp)*. *funcindex* determines which function will be computed and *obsindex* gives the "number" of the observation (≥ 1 and $\leq nob$).

objfcn.f

This is a subroutine that computes the objective function, given the model predictions and experimental observations. It has the following call sequence.

```
subroutine objfcn (cpredict,exptdata,fvec,nob)
```

Here *cpredict* is the model prediction, *exptdata* is the experimental observation and *fvec* is computed as a measure of deviation between *cpredict* and *exptdata*.

predictedc

After a successful regression, the model predictions and the observation times that they correspond to are printed to this file in two columns. The first column is the times and the second is the predictions.

fitparamlist

As the search through the parameter space continues, this file stores each set of parameters tested out and the corresponding sum of squares of residuals.

3.6 Library Packages

RAFT uses several FORTRAN codes on public domain. These are the packages BLAS, LINPACK, NSPCG for linear algebra; ODEPACK, DASSL, RADAU5, for the solution of initial-value ODE/DAE systems; NL2SOL, DQED, MINPACK, for parameter estimation and NNES, INTBIS for the solution of nonlinear systems of equations. In addition the routines from SLATEC library for computing machine parameters are used. The following table lists the packages used by RAFT and indicates how to obtain them. All of the above packages except (RADAU5) can be obtained from Netlib by anonymous ftp. Connect to Netlib by "ftp netlib2.cs.utk.edu".

BLAS	Basic Linear Algebra Subroutines. From Netlib, in the directory /blas. This is needed by most other packages.
LINPACK	Subroutines used in the direct solution of linear systems. From Netlib, in the directory /linpack. This is also used by several other packages.
NSPCG	Subroutines used in the iterative solution of linear systems. From Netlib, in the directory /itpack. RAFT has direct calls to this package. The name of one of the subroutines "prep.f" is changed to avoid multiply declared object files (ODEPACK has a routine by that name).
ODEPACK	Subroutines for the solution of initial-value Ordinary Differential Equations or Differential-Algebraic Equations. From Netlib, in the directory /odepack. The routines LSODA.f and LSODI.f are used by RAFT.
DASSL	A program for the solution of initial-value Ordinary Differential Equations or Differential-Algebraic Equations. From Netlib, in the directory /ode. Directly called by RAFT.
RADAU5	A program for the solution of initial-value Ordinary Differential Equations or Differential-Algebraic Equations. Available by anonymous ftp from "ftp.unige.ch", in the directory "pub/doc/math/stiff". Directly called by RAFT.
NNES	Nonmonotonic Nonlinear Equation Solver. From Netlib, in the directory /opt. Called by RAFT with small modifications to the code.
INTBIS	A program for solving nonlinear systems of equations. From Netlib, in the directory /toms (# 681). Called by RAFT with small modifications to the code. One of the subroutines "roots.f" is renamed to avoid multiply declared object files (ODEPACK has a routine by the same name).
DQED	A program for nonlinear regression by least squares. From Netlib, in the directory /opt. Directly called by RAFT.
NL2SOL	A program for nonlinear regression by least squares. From Netlib, in the directory /port (dn2gb.f). Directly called by RAFT.
MINPACK	Subroutines for minimization. From Netlib, in the directory /minpack. The subroutines LMDIF.f and FDJAC2.f are used by RAFT

MACHPARM

Subroutines D1MACH.f, R1MACH.f and I1MACH.f, to set machine parameters. From Netlib, in /slatec/src. These routines are used by almost all the packages.

3.7 Summary

The use of RAFT has been described in this chapter. The specification of the input data, input files (if necessary) and the interpretation of output are covered. The limitations/capabilities of RAFT can be gleaned from this information. They are reviewed here briefly.

- RAFT can solve problems in one-, two-, and three-dimensions in X-Y-Z cartesian space. The problem domain is restricted to be a cuboid. A suite of solvers for transport, reactions and regression are available so that a combination of numerical methods best suited for a problem can be chosen.
- The input files needed and the output files created by RAFT are listed here.

Input Files

adsorb

When adsorption is nonuniform for any of the components, this file supplies this nonuniform field of retardation factors.

columndata

In the absence of sources/sinks, injection normal to a face ($X = 0$) can be controlled by this file.

computepredict.f

If RAFT is to be used for parameter estimation then this subroutine computes the the predictions of the model (from a concentration profile) to be compared against observations.

conc-restart

If the initial concentration is nonuniform for any of the components then this file specifies the spatial distribution for all the components at the initial time.

data

This provides the most basic information about the problem and the numerical method to be used. This is needed at all times.

datarxn

If reactions are present then this file selects the methods for equilibration and ODE/DAE solution and specifies the reaction constants.

grid

If the grid is nonuniform in any one of the x-, y-, and z-directions then this file gives the grid plane locations for each of the directions.

lsqdata

If RAFT is to be used for parameter estimation then this file specifies which parameters are to be estimated and selects the method for nonlinear regression.

lsqexptdata

If RAFT is used for parameter estimation then this file gives the experimental observations.

objfcn.f

If RAFT is used for parameter estimation then this subroutine specifies the objective function to be minimized by the nonlinear regression procedure.

permx

If the hydraulic conductivity in x-direction k_x^h is nonuniform (when $ipermx = 1$) then this file supplies the field.

permy

If the hydraulic conductivity in y-direction k_y^h is nonuniform and is different from k_x^h (when $ipermy = 1$), then this file supplies the field.

permz

If the hydraulic conductivity in z-direction k_z^h is nonuniform and is different from k_x^h (when $ipermz = 1$), then this file supplies the field.

preeqrns.f

When reactions are present, and the reaction system consists of both kinetic and equilibrium reactions then this file of subroutines defines the nonlinear system of equations to be solved for equilibration after transport, before the solution of the differential-algebraic system.

rxns.f

When reactions are present, the user supplies appropriate subroutines that define the reaction system. Even in the absence of reactions these subroutines (perhaps with an empty body) must be present for successful compilation.

void

If porosity is nonuniform then this file supplies the field.

welldata

If sources/sinks are present then this file specifies the well configuration and controls injection.

xvelinput

This needed if the velocity field is being supplied instead of requiring RAFT to compute it.

yvelinput

Similar to **xvelinput**.

zvelinput

Similar to **xvelinput**.

Output Files

<u>Cprinttimes</u>	Gives the values of "time" at which the concentrations are printed out to the file <u>conc</u> .
<u>conc</u>	Gives the concentration profiles at selected time intervals.
<u>fitparamlist</u>	Gives the sequence of parameter sets tried out before arriving at the optimal set.
<u>histconc</u>	Gives the concentration history or breakthrough curves at selected spatial locations or YZ cross sections.
<u>masbal</u>	Gives information about mass balance for one of the components <i>masbalcomp</i> as the simulation progresses.
<u>potentials</u>	Gives potentials.
<u>predictedc</u>	Gives the final model predictions when the optimal set of parameters are used.
<u>wellrates</u>	Gives the flow-rates in/out of wells if wells are present. In the absence of wells, (a) it gives the flow-rate obtained, for the prescribed head drop along the x-direction; (b) it gives the inlet ($x = 0$) potential needed, for the prescribed flow-rate along the x-direction.
<u>xvel</u>	Gives the x-velocities.
<u>yvel</u>	Gives the y-velocities.
<u>zvel</u>	Gives the z-velocities.

- User specified source/sink configuration (including partially penetrating wells) can be handled. A linear displacement type problem can be solved (in the absence of wells) by setting the number of wells to zero.

- Arbitrary injection schemes can be handled. That is, wells, component injection can be turned on/off at user specified times.
- Time-step sizes are determined by the nominal size specified in the input `data`, injection strategy and the numerical method. In the case of wells, the time-step size should be such that the amount of fluid injected into a well-block is not greater than the volume of that block and the amount of fluid extracted from a production well-block is not greater than the volume of that block.
- RAFT can solve mixed kinetic/equilibrium reaction systems with flow and transport. Reactions are input to the code through a user supplied FORTRAN file `rxns.f` and `preeqrns.f`. Preparation of these files for arbitrary sets of reactions is automated through a symbolic computational language MAPLE. A variety of solvers are available for the initial-value differential-algebraic problem and for the solution of nonlinear system for equilibration.
- RAFT can be used to estimate process parameters from field/laboratory experimental observations. This feature can also be used to select the source/sink configurations, flow rates and nutrient/chemical injection strategies for an optimal design of a remediation system.
- Three different methods are used in the solution of the advection-dispersion-reaction equation (Equations 2.1 through 2.5d). The first method is based on the Modified Method of Characteristics (CCM/CMM). The second one is based on a Flux-Corrected Transport algorithm (FCT). The third method is a total variation diminishing scheme (TVD). All three methods treat reactions with operator-splitting. FCT and TVD methods have more in common with each other than they do with CCM/CMM. A few important differences between FCT, TCD and CCM/CMM and their implementations in RAFT should be noted.

1 In CCM, the potentials and velocities are functions (though weak) of the time-step size, porosity and retardation. So in the first method, the velocities are recomputed whenever the time-step changes. If potentials are the goal of the simulation then a very small time-step size, unit porosity, and unit retardation should be used.

A cell-centered finite difference approach is used for velocities when FCT/TVD is the method of solution for concentrations. These velocities are independent of the time-step size, porosity and retardation.

- 2 In CCM, the velocities (x-, y-, and z-) are computed at 26 locations on the surface of a grid block.

- 3 In the absence of source/sinks the time-step size for CMM is determined by the requirement of non-intersecting streamlines. When sources/sinks are present then the allowable time-step size is limited by the volumes of the well-blocks. The amount of fluid injected/extracted from any well-block should be less than the volume of that well-block.

- 4 Nonuniform grid is allowed with CCM/CMM.

- 5 *method-0* (CCM/CMM) is usually much more CPU intensive. Most time is spent in integrations involving twisted cells. Also for small Courant numbers CCM/CMM can show large numerical dispersion. CCM/CMM is best used for problems that allow large time-step sizes (Courant number > 1.0).

For FCT/TVD each (x-, y-, z-) velocity is computed at just two locations; the centers of the two opposite faces of the grid block. Figure 3.2 shows the locations where x-, y-velocities are defined on a sample grid for the two methods.

FCT/TVD requires in all cases that the Courant number be less than 0.5 or even smaller. When sources/sinks are present then the allowable time-step size is limited by the volumes of the well-blocks. The amount of fluid injected/extracted from any well-block should be less than the volume of that well-block. Courant number is taken to be the maximum of,

$$\frac{V_x \Delta t}{\phi \Delta x}, \frac{V_y \Delta t}{\phi \Delta y}, \frac{V_z \Delta t}{\phi \Delta z},$$

where V is the Darcy velocity.

In the current version of RAFT, FCT needs the grid to be uniform. But TVD allows nonuniform grid.

method-1 (FCT) or *method-2* (TVD) is the recommended method for most problems of interest. FCT/TVD is an **order of magnitude** faster than (CCM/CMM), for most problems, especially in multidimensions (≥ 2).

Choosing the solvers to use Of the three solvers for the advection problem, FCT & TVD (*method-1* and *method-2*) are fast. TVD is pretty robust. When *order* = 0, FCT also is robust, but shows excessive numerical dispersion compared to TVD. When *order* = 2 or 4, FCT can go unstable and give oscillatory solutions for some problems. If it does not go unstable, FCT with *order* = 2 or 4, can preserve sharp fronts (low numerical dispersion). CCM/CMM is best used for problems that allow large time-step sizes so that the resulting savings in CPU time outweigh the inherent cost of computation for a single step. If time-step size has to be limited anyway because of fast reactions then TVD or FCT may be best scheme for the problem.

Several solvers are provided for the reaction problem. Depending on the type of reactions and their stiffness, appropriate solvers should be chosen. MAPLE program

Figure 3.3: Choosing a Solver for Reactions.

to generate the subroutines `rxns.f` and/or `preeqrxns.f` may be used if the reaction system is complex. Figure 3.3 illustrates the selection of appropriate solver(s) for a given reaction model. For example if we have a differential-algebraic reaction system then INTBIS or NNES can be used for equilibration and DASSL or RADAU5 or LSODI can be used for the solution of initial-value DAE problem. Though RADAU5 can solve DAE systems with index upto three, the current version of RAFT requires the reaction system to have unit index. Solution of DAE systems obtained by replacing very fast reactions with equilibrium relations, can sometimes be very CPU intensive and impractical in the presence of transport (on the other hand the DAE systems obtained by replacing some dependent ODES with conservation laws, are just about as difficult/easy if not easier to solve as the original ODE system). Small enough time-step sizes may be necessary for transport so that the solution after transport is *not too far* from equilibrium. This is demonstrated in an example in the following chapter. In such cases, it may be useful to treat the equilibrium reactions as kinetic reactions with large rate constants. Larger and larger rate constants may be used until the solution is independent of those rate constants (and until *all* the solvers *break* due to increasing stiffness). If such a converged solution is obtained then that solution is usually quite close to the solution obtained by a the equilibrium formulation of the fast reactions, but cheaper. If such a converged solution is not obtained then the actual equilibrium formulation has to be solved.

Of the three methods available for optimization, DQED and NL2SOL are probably quite robust. DQED is usually faster. LMDIF does not allow lower/upper bounds on the parameters and so has limited use.

RAFT as an Engineering Design Tool As an engineering design tool, RAFT can be used for solving a variety of problems in the transport of groundwater contaminants. The flexibility of the code allows it to model all the way from shake-flask experiments in the laboratory to extensive field experiments. A typical sequence (by no means exhaustive) of problems faced in the development of a groundwater remediation technology (for example, bioremediation), and the use of RAFT at each step are listed here. RAFT has been used at almost all the steps of this sequence on some problem or other.

- Use RAFT to determine the mechanism (either empirical or fundamental) for the reactions occurring in a shake-flask. A variety of reaction models can be quickly incorporated into RAFT through the use of the MAPLE program. Parameter estimation tools can be used to determine the reaction parameters in each case from the experimental data.
- Use RAFT to model the results from column experiments. Determine the dispersivities from the tracer breakthrough. Extend the reaction model to capture any additional processes present under transport. Confirm the reaction model and the reaction parameters obtained from batch data, through an independent estimation of the reaction parameters from the column data. A few iterations between the batch and column experiments will yield a single parametric set that describes the reactions. Sensitivity of the model predictions to the fitted parameters has to be established.
- Use RAFT to study the effects of physical and chemical heterogeneities on contaminant transport. Intermediate scale experiments with known heterogeneities can be designed and simulated, as a prelude to the field scale experiment. The reaction model can be further refined to correctly predict the effects of heterogeneities.
- With all the site characterization data in hand, RAFT can be used to design the remediation experiment in the field. Tracer data can be used to extract some information on the hydraulic conductivity and the dispersivities. The use of geostatistics coupled with sensitivity analysis can help establish worst/best case events. Optimal design can be attempted at this time to select the operating parameters of the remediation process. For example in a bioremediation experiment, the injection concentrations and the pulsing strategy for nutrients, and the well configuration could be optimized.

Chapter 4

Examples

A detailed description of RAFT was given in Chapter 3. Here the preparation of the input data for a few sample problems is illustrated. The output from RAFT is interpreted and analyzed to determine the quality of simulation. These should help guide the preparation of input data and interpretation of output for any other problem. Also RAFT is validated by demonstrating convergence to known analytic solutions for simple problems. As mentioned in Chapter 3, the input data file `welldata` has a text header that lists the variable names in the same order as the entries that follow. For files `data` and `columndata`, and `datarxn` the names of the variables on each card appear just before they are entered.

4.1 Tracer Transport through a Column

Consider the advective-diffusive transport of a nonreactive tracer through a column. The geometry and the flow boundary conditions are sketched in Figure 4.1. A step input of tracer is applied at the face $x = 0$. The units are chosen to be meters and days.

4.1.1 Constant injection

Here the steady, constant injection of tracer (step input) at the $x = 0$ face is considered.

Input Since there are no sources/sinks the input data file `welldata` need not be prepared. Like wise since there are no reactions the files `datarxn` and `rxns.f` need not be prepared. No input files are necessary since ϕ , R_i , k^h and the initial concentration are taken to be uniform. Only the input data files `data` and `columndata` are needed. Both the files can be easily prepared by consulting Sections 3.1.1 and 3.1.2. A typical `data` file for this problem is prepared here.

`data`

Card # 1	We choose to use <i>method</i> = 0. The second entry is not used. We are not estimating parameters, so the third entry <i>lsqfit</i> = 0.	0 4 0
----------	---	-------

Card # 2	There is a single component. So $ncomp = 1$.	1
Card # 3	The problem is one-dimensional. So only the X-extent needs to be discretized. L_x is divided into 100 sections. So $nxp = 100$, $nyp = nzp = 1$.	100 1 1
Card # 4	Simulation starts at time 0.0. We want the simulation results till 0.5 days, using a time-step size not greater than 0.01 days. The fourth entry is not used here. So $tstart = 0.0$, $tfinal = 0.5$ and $\Delta t = 0.01$.	0.0 0.5 0.01 0
Card # 5	The simulation is starting at a time 0.0 days with a uniform initial concentration. So $restart = 0$.	0
Card # 6	Conductivity, adsorption and porosity are uniform. Also only k_x^h is of interest in this problem. So $ipermx = 0$, $ipermy = ipermz = -1$ and $iadsorb = iporos = 0$. We want the code to solve for velocities, so $donotsolvevel = 0$.	0 0 -1 -1 0 0
Card # 7	The number of tracetypes is equal to 1 since there is no adsorption. So $ntracetypes = 1$.	1
Card # 8	The grid is uniform. So $nonungrid = 0$.	0
Card # 9	There are no reactions, so $rxnflag = 0$. In one-dimension the dispersion tensor is always diagonal, so $dispdia = 0$ or 1. Volumes always match in one dimension. So $forcevolbal$ is not used here.	1 0 1
Card # 10	We want the concentration to be output after every 0.1 days from the start.	0.1
Card # 11	We want the code to cut the time-step if necessary, but not more than 8 times. So $maxtcut = 8$. We want the code to report the mass balance summary for the component 1. So $masbalcomp = 1$. We use a one-step forward Euler integration. So $ntracksteps = 1$.	8 1 1
Card # 12	We allow a maximum of 100 iterations for the concentration problem and want the residual reduced by a factor 10^{-20} . We allow a maximum of 50 iterations for the velocities and want the relative error in volumes to be less than 10^{-7} .	100 1.0e-20 50 1.0e-7
Card # 13	The domain has a unit length (1.0 meters) in X-, Y- and Z-directions. So $L_x = L_y = L_z = 1.0$.	1.0 1.0 1.0

Card # 14	Porosity ϕ is taken to be equal to 1.0.	1.0
Card # 15	The viscosity of the flow does not change with concentration. So $M = 1.0$.	1.0
Card # 16	The conductivity in the X-direction (k_x^h) is taken to be equal to 1.0 meters/day. k_y^h and k_z^h are immaterial to this problem.	1.0 1.0 1.0
Card # 17	Molecular diffusion is being neglected here. The longitudinal dispersivity (α_l) is taken to be 0.002 meters. α_t is immaterial to this problem.	0.0 0.002 0.0002
Card # 18	As shown in Figure 4.1 we impose a potential gradient between $x = 0$ and $x = L_x$. No-flow condition is employed on all the other faces. We use a zero dispersive flux condition at outflow boundaries (here only $x = L_x$), so $cbc = 1$.	1 1 0 0 0 0 1
Card # 19	The potential at $x = 0$ is specified to be 1.0 meters and at $x = L_x$, it is set to 0.0 meters. The other entries do not matter.	1.0 0.0 0.0 0.0 0.0 0.0
Card # 20	We want the potentials, velocities to be printed out (in addition to of course the concentrations). So $presflag = velflag = 1$. We also want output to screen. So $outputflag = 1$.	1 1 1
Card # 21	There are no wells in the problem. So $iwells = pwells = 0$.	0 0
Card # 22	The domain is initially devoid of the tracer. So $initconc(1) = 0.0$.	0.0
Card # 23	There is no adsorption. So $stick$ will be forced to 0.	0
Card # 24	There is no adsorption. So $nustick$ will be forced to 1.0.	1.0
Card # 25	All the components(= 1) here are under the same $tracetype(= 1)$. There are no immobile components. So $ntracecomp(1) = 1$ and $nimmobilecomp = 0$.	1 0
Card # 26	The components(= 1) under each $tracetype(= 1)$; $compId(1,1) = 1$.	1

Card # 27	The minimum possible concentration is 0.0 and the maximum is 1.0.	0.0 1.0
Card # 28	We do not want the history of concentrations printed out. So <i>nhistlocs</i> = 0. Since <i>nhistlocs</i> = 0, the other entries do not matter.	0 1 1 0.1
Card # 29	Since <i>nhistlocs</i> = 0, this card is not read.	1 1 1
Card # 30	Since <i>nhistlocs</i> = 0, this card is not read.	1

columndata

Card # 1	Since the fluid is being injected at a face $x = 0$, <i>injfaceflag</i> = 1.	1
Card # 2	We are not imposing a constant flow-rate. So <i>flowflag</i> = 0. The second entry does not matter in this case.	0 1.0
Card # 3	Component # 1 is being injected at this face.	1
Card # 4	We have a step input of concentration. So the number of pulses is 1.	1
Card # 5	Injection is uniform across the $x=0$ face. So <i>nonuniforminj</i> = 0.	0
Card # 6	This card will not be read since <i>nonuniforminj</i> = 0.	1 1 1 1
Card # 7	The injection of the component starts at time 0.0 days and ends at the final time 0.5 days. The concentration of the pulse is 1.0 grams/(meter ³).	0.0 0.50 1.0

Output This simple problem has analytic solutions to both velocities and concentrations. Velocities are given by,

$$\begin{aligned}
 V_x &= 1.0, \quad V_y \equiv V_z \equiv 0.0, \\
 \psi(x) &= \psi(x=0) - xV_x/k_x^h.
 \end{aligned}
 \tag{4.1}$$

Concentrations are given by [32],

$$C(x,t) = \operatorname{erfc} \left[\frac{(x-t)}{\sqrt{Dt}} \right] + \left[\frac{t^{1/2}}{\pi D} \right] \exp \left[-\frac{(x-t)^2}{4Dt} \right] + \frac{1}{2} \left[1 + \frac{x}{D} + \frac{t}{D} \right] \exp \left[\frac{x}{D} \right] \operatorname{erfc} \left[\frac{x+t}{2\sqrt{Dt}} \right]. \quad (4.2)$$

The numerical solution from RAFT can be compared to that of the above analytic solution. The output file potentials contains 100 numbers, one value of potential at the center of a grid cell, for each grid cell, going from $x = 0.005$ to $x = 0.995$. Following is a part of the contents of the file potentials.

```
.995000000000E+000 .985000000000E+000 .975000000000E+000
.965000000000E+000 .955000000000E+000 .945000000000E+000
```

```
.350000000000E-001 .150000000000E-001 .500000000000E-002
```

It is easily verified that the numerical solution for potentials matches with the analytic solution. It is noted from the files xvel, yvel and zvel that the velocity solution also matches. The file conc contains the concentration profile of the tracer at some times. The file Cprintimes lists what these times are. Since *timeprint* has been chosen to be 0.1 days, concentrations at the centers of 100 grid cells are printed at the following times (in days, from the file *Cprintimes*).

```
0.999999999999999917E-01
0.2000000000000000039
0.30000000000000000100
0.40000000000000000189
0.50000000000000000000
```

Figure 4.2 shows the numerical and the analytical solutions at the above times and they are seen to be nearly identical. The file masbal has 50 lines (since there were 50 time-steps). The entries diffcbal and cumcbal in each line are very close to zero (the small error is due to numerical roundoff/truncation errors). Thus we conclude that the numerical simulation preserved mass. We note from this file that, at the final time 0.5 days, 0.5 units of tracer is present in the column. We expect this because the total amount of tracer injected is

$$V_x * (t_{final}) * (c^0) * L_y * L_z = 1.0 * 0.5 * 1.0 * 1.0 * 1.0 = 0.50.$$

Also from the concentration profile at t_{final} that shows zero concentration for tracer at the exit $x = 1.0$, we expect no breakthrough. The file masbal confirms this.

Figure 4.1: Geometry and boundary conditions for transport through a column.

Figure 4.2: Concentration profiles. At five different times (0.1, 0.2, 0.3, 0.4 and 0.5 days), the numerical solution (circles) matches with the analytic solution (solid line). *method* = 0.

Figure 4.3: The numerical solution has a linear convergence with respect to Δt . *method* = 0.

convergence Figures 4.3, 4.4, and 4.5 show that the root mean square value of the error $\int_{x=0}^1 (C_{analytic} - C_{numerical})^2 dx$ decreases as Δt or Δx or both decrease. Simulations are performed with a large n_{xp} ($=10000$), $\alpha_l = 0.2$ meters, and a series of Δt , $\{0.04, 0.02, 0.01, 0.005, 0.0025\}$. In Figure 4.3 $\log(\int_{x=0}^1 (C_{analytic} - C_{numerical})^2 dx)$ is plotted against $\log(\Delta t)$, at a few times. The slope of the lines is approximately 1.0, giving us a unit rate of convergence with Δt . To show convergence with Δx , simulations are performed with a series of $\Delta x = 1.0/n_{xp}$, $\{0.02, 0.01, 0.005, 0.0025\}$ and a Δt chosen to be equal to Δx^2 . Again $\alpha_l = 0.2$ meters. Figure 4.4 plots $\log(\int_{x=0}^1 (C_{analytic} - C_{numerical})^2 dx)$ against $\log(\Delta x)$ at several times. The slope of the lines is approximately 1.9 - 2.0. So the convergence with grid size is almost quadratic. Figure 4.5 shows a similar plot for a much smaller dispersion ($\alpha_l = 0.005$ meters). Again a near quadratic convergence is obtained.

The above problem is also solved with *method* = 1 and *method* = 2. Figures 4.6 and 4.7 show that these numerical solutions also match the analytic solution.

Figure 4.4: The numerical solution has a near quadratic convergence with respect to Δx . High dispersion. *method* = 0.

4.1.2 Pulsing square wave. Pure advection

In this problem the tracer is pulsed through the column at varying concentrations. Dispersion is taken to be zero. This is a tougher problem to solve than the one in the previous section. The difficulty arises because of the multiple steep/narrow fronts that have to propagate without dissipation. Most numerical schemes result in a spreading of the front due to numerical dispersion. The file `data` to this problem is almost the same as before. Card # 17 is modified.

Card # 17 For a pure advection problem, $d_m = 0.0$, $\alpha_i = 0.0$ and $\alpha_t = 0.0$ 0.0 0.0 0.0

The file `columndata` is changed to specify the pulsing strategy. There are 10 pulses of varying width and height (concentration).

`columndata`

Card # 1 Since the fluid is being injected at a face $x = 0$, *injfaceflag* = 1. 1

Card # 2 We are not imposing a constant flow-rate. So *flowflag* = 0. The second entry does not matter in this case. 0 1.0

Figure 4.5: The numerical solution has a near quadratic convergence with respect to Δx . Low dispersion. *method* = 0.

Card # 3	Component # 1 is being injected at this face.	1
Card # 4	We are injecting 10 pulses of component 1. So <i>Fnswitchcomp</i> (1) = 10.	10
Card # 5	Injection is uniform across the <i>x</i> =0 face. So <i>nonuniforminj</i> = 0.	0
Card # 6	This card will not be read since <i>nonuniforminj</i> = 0.	1 1 1 1

Figure 4.6: Concentration profiles at 0.1, 0.2, 0.3, 0.4 and 0.5 days. The numerical solution (circles) matches with the analytic solution (solid line). *method* = 1.

Cards 7
thru 16

The start/end times, and the concentration of each pulse is given here. Each card contains *Fcompstarttime*, *Fcompendtime*, *Foriginjconc* of a pulse.

0.02	0.07	0.1
0.08	0.12	0.3
0.13	0.19	0.5
0.21	0.25	0.7
0.25	0.28	0.9
0.28	0.32	0.9
0.33	0.37	0.7
0.38	0.42	0.5
0.43	0.47	0.3
0.48	0.50	0.1

Figures 4.8 and 4.9 show the breakthrough curve at the exit of the column. Figure 4.8 shows simulation result with $n_{xp} = 100$ and the analytic solution. Figure 4.9 shows the numerical solutions with $n_{xp} = 50$ and $n_{xp} = 150$. The analytic solution which is identical to the input injection is matched exactly by the simulation with $n_{xp} = 100$. The integrity of each pulse is maintained throughout its passage through the column. Smeared pulses are observed for $n_{xp} = 50$ and $n_{xp} = 150$. This is because Courant number ($V\Delta t/\Delta x$) is an integer (1 in this case) only when $n_{xp} = 100$. Small numerical dispersion may be observed if Courant number is not an integer. This numerical dis-

Figure 4.7: Concentration profiles. At five different times, the numerical solution (circles) matches with the analytic solution (solid line). *method* = 2.

persion can be decreased by using a finer grid size. The solution with $n_{xp} = 150$ is less smeared and closer to the analytic solution than the solution with $n_{xp} = 50$.

Figures 4.10 and 4.11 show the solution with *method* = 1, with the order of higher order fluxes being 2 and 4 respectively. The use of fourth order fluxes generates less numerical dispersion. Figure 4.12 shows the solution with *method* = 2.

4.2 Reactive Transport in One-Dimension

Here a reactive tracer that undergoes a first order reaction is introduced at the entry ($x = 0$) of the column. Otherwise the problem is identical to the problem in Section 4.1.1. If $r(C) = -k_d C$ then the analytic solution is given by [32],

$$\begin{aligned}
 C(x,t) = & \frac{1}{1+u} \exp \left[\frac{(1-u)x}{2D} \right] \operatorname{erfc} \left[\frac{(x-ut)}{2\sqrt{Dt}} \right] + \\
 & \frac{1}{1-u} \exp \left[\frac{(1+u)x}{2D} \right] \operatorname{erfc} \left[\frac{(x+ut)}{2\sqrt{Dt}} \right] + \\
 & \frac{1}{2k_d D} \exp \left[\frac{x}{D} - k_d t \right] \operatorname{erfc} \left[\frac{x+t}{2\sqrt{Dt}} \right], \quad (4.3)
 \end{aligned}$$

Figure 4.8: Analytic breakthrough curve is exactly matched by simulations with integer Courant numbers. *method* = 0.

Figure 4.9: Breakthrough. Finer grid size reduces numerical dispersion. *method* = 0.

Figure 4.10: Breakthrough. Finer grid size reduces numerical dispersion. *method* = 1, *order* = 2.

Figure 4.11: Breakthrough. Finer grid size reduces numerical dispersion. *method* = 1, *order* = 4.

Figure 4.12: Breakthrough. Finer grid size reduces numerical dispersion. *method* = 2.

where,

$$u = (1 + 4k_d D)^{1/2}.$$

The file **data** is similar to the one in Section 4.1.1. Card #9 is modified to,

Card # 9 In one-dimension the dispersion tensor is always diagonal, so *dispdia* = 0 or 1. We are 1 1 1
considering reactions, so *rxnflag* = 1.

so that *rxnflag* is turned on. We choose to use RK4 (*odeflag* = 0) for the ODE. So, the subroutine **defrxns** alone needs to be prepared in the file **rxns.f**. **defrxns** consists of just one line.

```
func(1) = - rxnk(1)*rxc(1)
```

rxnk(1) is the rate constant k_d . k_d is chosen as 0.25 day^{-1} . The file **datarxn** can be easily prepared.

datarxn

Card # 1 We choose to use the RK4 method. So *ode-*
flag = 0. 0

Card # 2	There are no algebraic constraints in the reaction system. So <i>nalgebraic</i> = 0. The other two entries do not matter.	0 0 2
Card # 3	No equilibrium reactions. So <i>eqcompId</i> = { }.	
Card # 4	Since there are no equilibrium reactions in the reaction system, there is no need for equilibration. These entries do not matter.	1.0d-16 20000 0 1.0d-6
Card # 5	Since there are no equilibrium reactions in the reaction system, there is no need for equilibration. These entries do not matter.	10000 0 1.0d-7 1.0d-10
Card # 6	Since there are no equilibrium reactions in the reaction system, there is no need for equilibration. Since <i>negpreeq</i> = 0, this card is empty.	
Card # 7	The following will not be used when <i>odeflag</i> = 0.	1.0e-8 1.0e-10
Card # 8	Two <i>RKsteps</i> are used in this example.	2
Card # 9	These are not used here.	0 1000
Card # 10	These are not used here.	0 100 1.0e-3 1.0e-2
Card # 11	These are not used here.	0 0 1
Card # 12	These are not used here.	0 1000
Card # 13	There is only one constant in the rate expression.	1
Card # 14	The value of <i>rnk</i> (1).	0.25

The number of *RKsteps* is chosen to be 2 in the above. Figure 4.13 shows the analytical/numerical solutions for the concentration profile in the column at various times. Figure 4.14 shows the analytical/numerical solution for the breakthrough curve. Numerical solution closely matches the analytic solution in both the figures. Figure 4.14 also shows that the numerical solution does not change much when 10 *RKsteps* are used. Reactions in this particular problem are not fast compared to flow. Damkohler

Figure 4.13: Concentration profiles at four different times. Numerical/Analytic solutions for a tracer undergoing 1st order reaction. *method* = 0.

number ($k_d L_x / V_x$) for this problem is 0.25. When reactions are much faster, the numerical solution of the problem may require small Δt as well as large number of *RKsteps*. $\Delta t / \text{RKsteps}$ is the time-step size used by the 4th order Runge-Kutta algorithm. Depending on the Damkohler number, appropriate Δt and *RKsteps* will have to be chosen. The other four methods can also be used for integrating the reactions.

Figures 4.15 and 4.16 show the breakthrough curves for this case obtained with *method* = 1 (and *order* = 4) and *method* = 2. Again the analytic solution is matched by both the numerical methods.

4.3 Reactive Transport in Two-Dimensions

Here the reactive flow and transport of multiple components through a heterogeneous medium in two-dimensions (X-Y cartesian space) is considered. The reactions are those given in Example 2 in Section 3.4.5. Oxygen rich water is injected normal to the face $x = 0$, into the domain initially devoid of oxygen. Hydrocarbon is initially present at a uniform concentration. Initial microbial population is nonuniform, varying randomly between 0.02 – 0.03 grams/meter³. The adsorption of hydrocarbon to the soil is given by a retardation factor R_h that varies in space. Microbes are taken to be immobile.

Figure 4.14: Numerical/Analytic breakthrough curves for a tracer undergoing 1st order reaction. *method* = 0.

Figure 4.15: Numerical/Analytic breakthrough curves for a tracer undergoing 1st order reaction. *method* = 1, *order* = 4.

Figure 4.16: Numerical/Analytic breakthrough curves for a tracer undergoing 1st order reaction. *method* = 2.

Hydraulic conductivity k^h varies in space. Following is the data to the problem.

$L_x = 50.0, L_y = 10.0, L_z = 1.0$ meters, Flow-rate = 1.0 meter³/day, $\phi = 0.50, \alpha_l = 0.2$ meters, $\alpha_t = 0.01$ meters.

c_O^0 : Step input concentration for Oxygen: 40 grams/meter³.

c_H^I : Uniform initial concentration for Hydrocarbon: 6.0 grams/meter³.

A very coarse grid is used here so that the input files (**permx**, **conc-restart**, **adsorb**) will be short. Following are the data files. All units are in grams, meters and days.

data

Card # 1	We choose to use <i>method</i> = 0. So the second entry does not matter. The third entry is 0 since we are not estimating parameters.	0 4 0
Card # 2	There are 4 components: Hydrocarbon, Oxygen, Microbes and tracer. So <i>ncomp</i> = 4.	4
Card # 3	The problem is two-dimensional (X-Y cartesian space). So only the X-, Y-extents are discretized. L_x is divided into 5 sections and L_y into 3 sections. So <i>n_x</i> = 5, <i>n_y</i> = 3.	5 3 1

Card # 4	Simulation starts at time 0.0. So $tstart = 0.0$. We want the simulation results till a time, $tfinal = 250$ days, with a time-step size Δt of 0.5 days. <i>forcetimestep</i> is not used here.	0.0 250.0 0.5 0
Card # 5	Since the initial concentration of microbes is nonuniform, the concentration distribution of all the components has to be specified in the file <code>conc-restart</code> . So <i>restart</i> = 1.	1
Card # 6	We want the velocities to be computed. So <i>donotsolvevel</i> = 0. X-hydraulic conductivity (k_x^h) is nonuniform. So <i>ipermx</i> = 1. $k_y^h \equiv k_x^h$, so <i>ipermx</i> = -1. Since some of the components adsorb, <i>iadsorb</i> = 1. Since porosity is uniform <i>iporos</i> = 0.	0 1 -1 -1 1 0
Card # 7	<i>ntracetypes</i> = 2. See the following for explanation.	2
a Oxygen and tracer do not adsorb. These two components (2 and 4) belong to say tracetype, 1.		
b Hydrocarbon adsorbs. So this component (1) belongs to a different tracetype, 2.		
Card # 8	Since the grid is uniform, <i>nonungrid</i> = 0.	0
Card # 9	We consider only the diagonal part of the full dispersion tensor to avoid the small oscillations. So <i>dispdiag</i> = 1. Since we have reactions, <i>rxnflag</i> = 1. We want to force volume balance so as to achieve mass balance, so <i>forcevolbal</i> = 1.	1 1 1
Card # 10	We want the concentrations of all the components to be output every 50 days from <i>tstart</i> . So <i>timeprint</i> = 50.0.	50.0
Card # 11	We choose <i>maxtcut</i> to be 8. We want the mass balance summary reported for the hydrocarbon (component 1). So <i>masbalcomp</i> = 1. We use a one-step forward Euler integration. So <i>ntracksteps</i> = 1.	8 1 1
Card # 12	We allow a maximum of 100 iterations for the concentration problem and want the residual reduced by a factor 10^{-20} . We allow a maximum of 50 iterations for the velocities and want the relative error in volumes to be less than 10^{-7} .	100 1.0e-20 50 1.0e-7
Card # 13	$L_x = 50$ meters, $L_y = 10$ meters and $L_z = 1.0$ meters.	50.0 10.0 1.0

Card # 14	Porosity, $\phi = 0.5$.	0.5
Card # 15	The viscosity of the flow does not change with concentration. So $M = 1.0$.	1.0
Card # 16	Hydraulic conductivity is nonuniform. So these entries are not used. Conductivity is read from the file <code>permx</code> .	1.0 1.0 1.0
Card # 17	Molecular diffusion is being neglected here. The longitudinal dispersivity, $\alpha_l = 0.2$ meters and the transverse dispersivity, $\alpha_t = 0.01$ meters.	0.0 0.2 0.01
Card # 18	As shown in Figure 4.1 we impose a potential gradient between $x = 0$ and $x = L_x$. Noflow condition is employed on all the other faces. Zero dispersive flux condition is used at the outflow boundaries.	1 1 0 0 0 0 1
Card # 19	The potential at $x = 0$ is specified to be 1.0 meters and at $x = L_x$, it is set to 0.0 meters. The other entries do not matter.	1.0 0.0 0.0 0.0 0.0 0.0 0.0
Card # 20	We want the potentials, velocities to be printed out (in addition to of course the concentrations). So <i>presflag</i> = <i>velflag</i> = 1.	1 1 1
Card # 21	There are no wells in the problem. So <i>iwells</i> = <i>pwells</i> = 0.	0 0
Card # 22	Since the initial concentration is nonuniform, they are read from the file <code>conc-restart</code> . The entries on this card are not used.	6.0 0.0 0.02 0.0
Card # 23	Since <i>ntracetypes</i> = 2, <i>stick</i> is an array of length 2. Oxygen and tracer are of trace-type 1 that do not adsorb, so <i>stick</i> (1) = 0). Hydrocarbon of trace-type 2, adsorbs, so <i>stick</i> (2) = 1.	0 1
Card # 24	Since <i>ntracetypes</i> = 2, <i>nustick</i> is array of length 2. <i>nustick</i> (2) = 1.0, since the adsorption for hydrocarbon is nonuniform. This will be read from the file <code>adsorb</code> .	1.0 1.0
Card # 25	Since <i>ntracetypes</i> = 2, <i>ntracecomp</i> is an array of length 2. <i>ntracecomp</i> (1) = 2 since two components, Oxygen and tracer are under trace-type 1. <i>ntracecomp</i> (2) is equal to 1 since trace-type 2 has one component (hydrocarbon) under it. The number of immobile components <i>nimmobilecomp</i> is 1 (microbes).	2 1 1

Card # 26	Oxygen is component # 2 and tracer is component # 4. So $compId(1,1) = 2$ and $compId(2,1) = 4$. Hydrocarbon is component # 1. So $compId(1,2) = 1$. Microbes is component # 3. So $immobilecomp(1) = 3$.	2 4 1 3
Card # 27	The minimum possible concentration for Oxygen, hydrocarbon and the tracer is 0.0. Microbes are not allowed to fall below a minimum concentration equal to 0.02. There is no upper bound on the concentration of microbes, so the maximum is taken to be a large value. The maximum for Oxygen is 40.0 since that is the injection concentration and can only decrease due to reaction. Similarly the maximum concentration for the tracer is 1.0. For the hydrocarbon the maximum concentration is its maximum initial concentration.	0.0 6.0 0.0 40.0 0.02 1.0e20 0.0 1.0
Card # 28	History of concentration is not needed. So $nhistlocs = 0$.	0 1 1 0.1
Card # 29	Since $nhistlocs = 0$, this card is not read.	1 1 1
Card # 30	Since $nhistlocs = 0$, this card is not read.	1
columnndata		
Card # 1	We are injecting at face $X = 0$, so $injfaceflag = 1$.	1
Card # 2	We control the flow-rate into the domain to be 1.0 meter ³ /day. So $flowflag = 1$ and $flowrate = 1.0$.	1 1.0
Card # 3	We are injecting Oxygen and tracer, components 2 and 4. So $Finject(1) = 0$, $Finject(2) = 1$, $Finject(3) = 0$ and $Finject(4) = 1$.	0 1 0 1
Card # 4	$Fswitchcomp$ is an integer array of length 4. Oxygen and tracer are injected continuously. So $Fswitchcomp(2) = 1$, $Fswitchcomp(4) = 1$.	0 1 0 1
Card # 5	Injection is uniform across the $x=0$ face. So $nonuniforminj = 0$.	0
Card # 6	This card will not be read since $nonuniforminj = 0$.	1 1 1 1

Card # 7	We have a step input of concentration 40.0 for Oxygen and 1.0 for the tracer. So $Fcompstartime(2) = Fcompstartime(4) = tstart = 0.0$ days, $Fcompendtime(2) = Fcompendtime(4) = tfinal = 250.0$ days, $Foriginjconc(2) = 40.0$, and $Foriginjconc(4) = 1.0$.	0.0 250.0 40.0 0.0 250.0 1.0
datarxn		
Card # 1	We choose to use the RK4 method. So $odeflag = 0$.	0
Card # 2	There are no algebraic constraints in the reaction system. So $nalgebraic = 0$. The other two entries do not matter.	0 0 2
Card # 3	No equilibrium reactions. So $eqcompId = \{ \}$.	
Card # 4	Since there are no equilibrium reactions in the reaction system, there is no need for equilibration. These entries do not matter.	1.0d-16 20000 0 1.0d-6
Card # 5	Since there are no equilibrium reactions in the reaction system, there is no need for equilibration. These entries do not matter.	10000 0 1.0d-7 1.0d-10
Card # 6	Since there are no equilibrium reactions in the reaction system, there is no need for equilibration. Since $neqpreeq = 0$, this card is empty.	
Card # 7	The following will not be used when $odeflag = 0$.	1.0e-8 1.0e-10 1.0e-8 1.0e-10 1.0e-8 1.0e-10 1.0e-8 1.0e-10
Card # 8	Twenty $RKsteps$ are used.	20
Card # 9	These are not used here.	0 1000
Card # 10	These are not used here.	0 100 1.0e-3 1.0e-2
Card # 11	These are not used here.	0 0 1
Card # 12	These are not used here.	0 1000

Card # 13 There are six constants in the rate expression. 6

Card # 14 The values for the six constants in appropriate units. 17.0 0.13 0.1
3.0 0.5 0.01

From the file **data** it is clear that the input files **permx** (since k_x is taken to be nonuniform), **conc-restart** (since the initial concentration is nonuniform) and **adsorb** (since R_h is nonuniform) are necessary.

permx A random distribution of conductivity between 0.0 and 1.0 meters/day is used. Conductivity for each grid block is shown in Figure 4.17. This translates into the following **permx** file to be read by the FORTRAN statements in Section 3.2.

0.8933	0.8816	0.3880	0.7504	0.4330
0.0626	0.5893	0.3811	0.2917	0.3646
0.7645	0.8481	0.8260	0.2762	0.8752

conc-restart Since at least one of the components has a nonuniform initial concentration, concentration distribution for all the components should be specified in this file. One value of concentration for each component is specified in each grid block.

6.0000	6.0000	6.0000	6.0000	6.0000
6.0000	6.0000	6.0000	6.0000	6.0000
6.0000	6.0000	6.0000	6.0000	6.0000
0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000
0.0278	0.0284	0.0260	0.0205	0.0223
0.0283	0.0254	0.0292	0.0281	0.0266
0.0207	0.0273	0.0210	0.0244	0.0267
0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000
0.0000	0.0000	0.0000	0.0000	0.0000

Lines 1-3 give the initial (uniform) concentration for component 1 (hydrocarbon), Lines 4-6 give the initial (uniform) concentration for component 2 (oxygen). Lines 7-9 give the initial (nonuniform) concentration for component 3 (microbes). Lines 10-12 give the initial (uniform) concentration for component 4 (tracer). For each component, these concentrations are mapped to the grid in the same fashion as the conductivity in Figure 4.17.

Figure 4.17: Conductivity mapped to the grid.

adsorb One of the components (hydrocarbon) adsorbs to the soil and the retardation factor R_h is nonuniform. R_h is taken to be randomly distributed between 1.0 and 10.0.

6.9936	8.9147	6.9572	1.9217	1.9793
9.4616	7.7800	8.9279	8.8993	1.0763
3.5819	5.1205	1.3439	3.5813	9.1790

The mapping of these values to the grid is as in Figure 4.17.

Since *masbalcomp* is set to 1, in the file **data**, material balance information for the hydrocarbon is output to the file masbal. Some of this information is plotted in Figure 4.18. When the coarse grid 5×3 is used, nearly 80% of the initially present hydrocarbon is consumed by the microbes in 250.0 days. With a finer grid 50×5 and with a different random distribution of conductivity and initial microbes only 60% is consumed. The larger degradation with the coarse grid is partly due to the larger numerical dispersion that widens the zone of mixing between oxygen and the contaminant. Since reactions occur in this mixing zone, a coarse grid predicts a larger degradation.

Figure 4.18: Microbial degradation of a hydrocarbon contaminant. *method* = 0.

4.4 Sources and Sinks

4.4.1 Single injection well

The preparation of the file `welldata` can be illustrated by a simple problem of a single fully penetrating injection well in a uniform medium. The solution to the problem should be radially symmetric. Also an input pulse at the well should match the breakthrough curve at some radial distance from the well in the case of zero dispersion. Following is the data to the problem. All units are in grams, meters and days.

$L_x = 31.0, L_y = 31.0, L_z = 10.0$ meters, Injection rate = $50.0 \text{ meter}^3/\text{day}$, $\phi = 0.60$, $\alpha_l = \alpha_t = 0.0$ meters, $c^I = 0.0$.

`data`

The well is at the center of the domain and a constant potential ($= 0.0$) condition is used on the X-Y boundaries. There is no flow through the top and bottom. Since there is no Z-component for velocity a two-dimensional (X-Y cartesian) problem can be solved. The nominal time-step size chosen is such that, the amount of fluid placed in a well-block is less than the volume of the well-block. Fluid injected during a time-step is $5 \text{ meter}^3 (=$

50.0 meter³/day $\times$ 0.1 days) and the volume of the well-block is equal to 6 meter³ (1 meter $\times$ 1 meter $\times$ 10 meters $\times$ 0.6).

Card # 1	We choose to use <i>method</i> = 0.	0 4 0
Card # 2	<i>ncomp</i> = 1.	1
Card # 3	X-extent and Y-extent are divided into 31 sections each. So <i>nzp</i> = <i>nyp</i> = 31.	31 31 1
Card # 4	<i>tstart</i> = 0.0. The time-step size, Δt is chosen to be 0.1 days. <i>tfinal</i> is 100 days. Since the time-step size is smaller than the maximum possible, we force the time-step size. <i>forcetimestep</i> = 1.	0.0 100.0 0.1 1
Card # 5	<i>restart</i> = 0, since the initial concentration is uniform ($\equiv$ 0.0).	0
Card # 6	We need to solve for velocities. So <i>donotsolvevel</i> = 0. Hydraulic conductivity is uniform, so <i>ipermx</i> = 0. $k_y^h \equiv k_x^h$, so <i>ipermy</i> = -1. There is no adsorption, so <i>iadsorb</i> = 0. Porosity is uniform so <i>iporos</i> = 0.	0 0 -1 -1 0 0
Card # 7	We have a single component so <i>ntracetypes</i> = 1.	1
Card # 8	We use a uniform grid. So <i>nonungrid</i> = 0.	0
Card # 9	We use the diagonal part of the full dispersion tensor, so <i>dispdia</i> = 1. There are no reactions, so <i>rxnflag</i> = 0. We want the volumes to match, so <i>forcevolbal</i> = 1.	1 0 1
Card # 10	We want the concentrations printed out to the file conc after every 1.0 days from <i>tstart</i> .	1.0
Card # 11	We allow the code to cut the time-step size for a maximum of 8 times. So <i>maxtcut</i> = 8. We want the code to report the mass balance summary for component # 1. So <i>masbalcomp</i> = 1. We use a one-step forward Euler integration. So <i>ntracksteps</i> = 1.	8 1 1
Card # 12	We allow a maximum of 100 iterations for the concentration problem and want the residual reduced by a factor 10^{-20} . These are not used in this example since a pure advection problem is being solved. We allow a maximum of 50 iterations for the velocities and want the relative error in volumes to be less than 10^{-7} .	100 1.0e-20 50 1.0e-7

Card # 13	$L_x = L_y = 31$ meters. $L_z = 10.0$ meters.	31.0 31.0 10.0
Card # 14	Porosity $\phi = 0.6$.	0.6
Card # 15	The viscosity of the flow does not change with concentration. So $M = 1.0$	1.0
Card # 16	The conductivity in the X-direction (k_x^h) is taken to be equal to 1.0 meters/day. k_y^h and k_z^h are equal to k_x^h .	1.0 1.0 1.0
Card # 17	This is a pure advection problem. So $d_m = \alpha_l = \alpha_t = 0.0$ meters	0.0 0.0 0.0
Card # 18	Constant potential condition is used on the X-, Y-boundaries. Noflow condition is used on the Z-boundaries. Zero dispersive flux condition is used at the outflow boundaries, so $cbc = 1$.	1 1 1 1 0 0 1
Card # 19	The values of the constant potential on the X-, Y-boundaries. The last two entries are not used here.	0.0 0.0 0.0 0.0 0.0 0.0
Card # 20	We want the potentials, velocities to be printed out (in addition to of course the concentrations). So $presflag = velflag = 1$.	1 1 1
Card # 21	We have a single injection well. So $iwells = 1$ and $pwells = 0$.	1 0
Card # 22	The initial uniform concentration of tracer is 0.0.	0.0
Cards 23 and 24	There is no adsorption. So $stick(1) = 0$ and $nustick(1) = 1.0$.	0.0 1.0
Card # 25	$ntracecomp = 1$ and $nimmobilecomp = 0$.	1 0
Card # 26	$compId(1,1) = 1$	1
Card # 27	The minimum possible concentration is 0.0 and the maximum is 1.0.	0.0 1.0

Card # 28 We want the history of concentrations at two locations in the grid. We want point values of concentration instead of averaged concentrations. So *bkthru* = 0. Also *nhistcomp* = 1 since we have only one component. We select *histime* = 0.1.

2 0 1 0.1

Card # 29 The locations of the two points is given here.

10 16 1
8 8 1

Card # 30 *histcomp*(1) = 1.

1

welldata

Pulses of tracer with different concentrations are injected. The well is located at the grid location (16,16), the center of the grid. The only cards needed are those for injection wells.

Card # 1 *nzp* = 1. So there is no splitting of flow into different layers. So *implicitrates* can be 0 or 1. We have no production wells and so no recirculation.

0 0

Card # 2 Injection well # 1 is at the grid location (16,16). It is fully penetrating. So *lociwell*(1,1) = 16, *lociwell*(2,1) = 16, *lociwell*(3,1) = 1 and *lociwell*(4,1) = 1.

16 16 1 1

Card # 3 The radius of the well *irw*(1) = 0.1 meters.

0.1

Card # 4 This injection well is switched on at *tstart* and shut off at *tfinal*. So *nswitchiwell*(1) = 1.

1

Card # 5 *iwellstarttime* = *tstart* = 0.0 days, *iwellendtime* = *tfinal* = 100.0 days. Injection rate, *origtotiqrate*(1) = 50.0 meter³/day.

0.0 100.0 50.0

Card # 6 Component 1 is being injected. So *Winject*(1,1) = 1.

1

Card # 7 The component 1 is being pulsed. There are 10 pulses. So *Wswitchcomp*(1) = 10.

10

Figure 4.19: 0.5 concentration contours with a single injection well.

Cards # 8
thru 17

The start/end times, and the concentration of each pulse is given here. *Wcompstarttime*, *Wcompendtime*, *Woriginjconc*.

0.0	10.0	1.0
10.0	20.0	0.8
20.0	30.0	0.6
30.0	40.0	0.4
40.0	50.0	0.2
50.0	60.0	0.2
60.0	70.0	0.4
70.0	80.0	0.6
80.0	90.0	0.8
90.0	100.0	1.0

Figure 4.19 shows the 0.5 concentration contour at 60 days to be symmetric around the injection point. The outer contour is close to the boundary and so is affected by the boundary condition. It is not as circular as the inner contour.

Figure 4.20 shows the breakthrough curves at three different radii ($r = 0.0, 6.0, 10.67$ meters). The input pulse is not as well preserved as in the case of column transport (Figure 4.8). There is more numerical dispersion in this case that smears the fronts. For small time-step sizes characteristics-based numerical methods show more numerical dispersion than when big time-step sizes are used. Since the time-step size here is forced to be less than or equal to the volume of a single grid block (well-block), a finer grid size

is required to preserve the fronts. A nonuniform grid with a large size for the well-block can also be used to increase the time-step size. Card # 3 in the above **data** file is modified to specify the finer grid size and Card # 8 is modified to indicate the use of a nonuniform grid.

Card # 2 $nzp = nyp = 61$. $nzp = 1$. 61 61 1

Card # 7 A nonuniform grid is being used. So *nonun-*
 grid = 1. 1

The file **grid** is necessary since *nonungrid* is set to 1. We choose the grid sizes so that the size of the well-block is $(1.0 \times 1.0 \times 10.0)$ and the remaining blocks is $(0.5 \times 0.5 \times 10.0)$.

grid

.000000	.500000	1.000000	1.500000	2.000000	2.500000
3.000000	3.500000	4.000000	4.500000	5.000000	5.500000
6.000000	6.500000	7.000000	7.500000	8.000000	8.500000
9.000000	9.500000	10.000000	10.500000	11.000000	11.500000
12.000000	12.500000	13.000000	13.500000	14.000000	14.500000
15.000000	16.000000	16.500000	17.000000	17.500000	18.000000
18.500000	19.000000	19.500000	20.000000	20.500000	21.000000
21.500000	22.000000	22.500000	23.000000	23.500000	24.000000
24.500000	25.000000	25.500000	26.000000	26.500000	27.000000
27.500000	28.000000	28.500000	29.000000	29.500000	30.000000
30.500000	31.000000				
.000000	.500000	1.000000	1.500000	2.000000	2.500000
3.000000	3.500000	4.000000	4.500000	5.000000	5.500000
6.000000	6.500000	7.000000	7.500000	8.000000	8.500000
9.000000	9.500000	10.000000	10.500000	11.000000	11.500000
12.000000	12.500000	13.000000	13.500000	14.000000	14.500000
15.000000	16.000000	16.500000	17.000000	17.500000	18.000000
18.500000	19.000000	19.500000	20.000000	20.500000	21.000000
21.500000	22.000000	22.500000	23.000000	23.500000	24.000000
24.500000	25.000000	25.500000	26.000000	26.500000	27.000000
27.500000	28.000000	28.500000	29.000000	29.500000	30.000000
30.500000	31.000000				
0.0	10.0				

Figure 4.20 shows that the breakthrough curves in this simulation are sharper.

Figure 4.20: Breakthrough curves at two different radii. *method* = 0

Figures 4.21 and 4.22 show the breakthrough curves with *method* = 1 (and *order* = 4) and *method* = 2. A uniform grid size is used for both fine and coarse grids.

4.4.2 A recirculation well

The use of RAFT to model partially penetrating wells is illustrated here. Consider a problem with a recirculation well. Figure 4.23 shows the well configuration. Production well reinjects into the injection well as shown in the figure. The wells are turned on/off a few times and also their flow-rates vary with time while they are on. Details can be found in the file `welldata` that follows. These features are included here to illustrate the use of RAFT in solving such problems. Tracer is injected into the domain initially devoid of any tracer. The medium is homogeneous and isotropic. The following is the data to the problem.

`data`

This file is nearly the same as the file `data` in Section 4.4.1. The following cards alone need to be changed.

Card # 1 We choose to use *method* = 2. No parameter 2 4 0
estimation.

Figure 4.21: Breakthrough curves at two different radii. *method* = 1, *order* = 4.

Figure 4.22: Breakthrough curves at two different radii. *method* = 2.

Figure 4.23: A dual screened recirculation well. Dual screened well is modeled as two partially penetrating wells at the same X-Y location.

Card # 2	There is only one component. $ncomp = 1$.	1
Card # 3	This is a problem in three dimensions. X-, Y- and Z-extents have to be discretized. Here $nxp = nyp = 31$, and $nzp = 10$.	31 31 10
Card # 4	$tstart = 0.0$, $tfinal = 150.0$ and $\Delta t = 0.04$ days. This Δt obeys the condition that the volume of fluid placed in a well-block is less than its volume. We require that the code not increase the time-step size, so $forcetimestep = 1$.	0.0 150.0 0.04 1
Card # 5	We need to solve for the flow field. So $donotsolvevel = 0$. The medium is homogeneous. There is no adsorption. Porosity is uniform.	0 0 -1 -1 0 0
Card # 13	$L_x = L_y = 31.0$ meters, $L_z = 30.0$ meters.	31.0 31.0 30.0
Card # 16	Hydraulic conductivity is equal to 1 M/day.	1.0 1.0 1.0
Card # 21	There is one injection well and one production well. So $iwells = 1$, $pwells = 1$.	1 1

Card # 28	We want the history of concentrations at four locations in the grid. We want point values of concentration instead of averaged concentrations. So $bkthru = 0$. Also $nhistcomp = 1$ since we have only one component. $histime = 0.04$.	4 0 1 0.04
		20 16 7
Card # 29	The locations of the two points is given here.	20 20 6
		20 20 5
		20 20 4
Card # 30	$histcomp(1) = 1$.	1
<div style="border: 1px solid black; padding: 2px; display: inline-block;">welldata</div>		
Card # 1	We use the implicit scheme to split the flow into layers. We have the production wells reinjecting into injection wells. So $reinjection = 1$.	1 1
Cards # 2 thru # 9 pertain to injection well # 1.		
Card # 2	Injection well # 1, is at the areal grid location (16,16) and is screened through the Z-grid blocks 6, 7, and 8. So $lociwell(1,1) = 16$, $lociwell(2,1) = 16$, $lociwell(3,1) = 6$ and $lociwell(4,1) = 8$.	16 16 6 8
Card # 3	The radius of the well $irw(1) = 0.1$ meters.	0.1
Card # 4	This injection well is switched on/off 5 times. So $nswitchiwell(1) = 5$.	5
		0.0 10.0 50.0
		20.0 30.0 5.0
Cards # 5 thru # 9	Each card indicates $iwellstarttime$, $iwellendtime$ and $origtotigrate$.	40.0 60.0 50.0
		60.0 70.0 5.0
		80.0 150.0 50.0
Card # 10	Tracer is injected at this well. So $Winject(1,1) = 1$.	1
Card # 11	There is single pulse of injection. So $Wnswitchcomp(2,1) = 1$.	1
Card # 12	Tracer is injected continuously from 0.0 days to 60.0 days at a concentration of 1.0. So $Wcompstarttime(1,1) = 0.0$, $Wcompendtime(1,1) = 60.0$, and $Woriginjconc(1,1) = 1.0$. Note that tracer is injected only when the well is ON.	0.0 60.0 1.0

Card # 13	Production well # 1 is at the areal grid location (16,16) and is screened through the Z-grid blocks 3, 4 and 5. So $locpwell(1,1) = 16$, $locpwell(2,1) = 16$, $locpwell(3,1) = 3$ and $locpwell(4,1) = 5$.	16 16 3 5
Card # 14	The radius of the well $prw(1) = 0.1$ meters.	0.1
Card # 15	Production well # 1 is switched on/off five times in an identical fashion as the injection. $well.nswitchpwell(1) = 5$.	5
Cards 16 thru 20	The five on/off times for this well and the flow-rate when it is on. Each card indicates $iwellstarttime$, $iwellendtime$ and $origtotigrate$.	0.0 10.0 50.0
		20.0 30.0 5.0
		40.0 60.0 50.0
		60.0 70.0 5.0
		80.0 150.0 50.0
Card # 26	When it is on, production well # 1 injects into injection well # 1. $p2iwellmap(1) = 1$.	1

Figure 4.24 shows the breakthrough curves at five different locations. At locations (16,16,7) and (20,16,7), the concentration goes up quickly for the first 10 days. This corresponds to the pumping at $50 M^3/day$ for the first 10 days. There is no pumping for the next 10 days so there is no flow. Since dispersion is absent the concentration remains unchanged all over the domain from day 10 through day 20. Later the concentration raises at a slower rate, corresponding to the smaller pumping rate of $5 M^3/day$. Since the tracer injection stops after 60 days, the concentration begins to fall. At later times some of the tracer will be lost through the boundaries. This loss and numerical dispersion will drive the concentration to zero all over the domain after sufficiently long times.

4.5 Coupled Reaction Systems

Here the solution of coupled kinetic and equilibrium reaction systems under batch conditions is first illustrated. As a first example consider the following simple system to which an analytic solution can be written.

Equation 4.4a is fast and can be considered to be at equilibrium. In that case the equilibrium relation is,

$$k_{eq1}a - b = 0 \quad (4.4c)$$

Figure 4.24: Tracer breakthrough curves at five locations for a series of on/off pulsing of a dual screened well. *method* = 2.

Let a_0 , b_0 , and c_0 be the initial values of a , b , and c . The solution to Equation 4.4b and Equation 4.4c is

$$a = a_0 e^{-(tk_{f2})/(1+k_{eq1})}, \quad (4.4d)$$

$$b = k_{eq1} a_0 e^{-(tk_{f2})/(1+k_{eq1})}, \quad (4.4e)$$

$$c = (a_0 + b_0 + c_0 - (1 + k_{eq1})a_0) e^{-(tk_{f2})/(1+k_{eq1})}. \quad (4.4f)$$

This system can also be solved by RAFT either as a mixed system, (with $N_e = 1$, $N_k = 1$, and $N_c = 1$) or as an all kinetic system (by the use of large k_{f1} and k_{b1}). The preparation of the file `rxns.f` follows the procedure described in Section 3.4. A batch system can be simulated by RAFT by choosing a zero potential gradient in all directions and by not having any sources/sinks. A few relevant cards from the files `data`, `columndata`, and `datarxn` are listed here.

`data`

Card # 1	Since there is no transport, either method can be chosen. No parameter estimation.	1 4 0
Card # 3	Since we have a batch system, one grid block sufficient.	1 1 1

Card # 17	Choose noflow in y-, and z-directions. Use constant potential conditions at $x = 0$ and $x = L_x$. Since there is no dispersion <i>cbc</i> is not used.	1 1 0 0 0 0 1
Card # 18	Use a zero potential gradient along the x-direction.	0.0 0.0 0.0 0.0 0.0 0.0 0.0
Card # 20	Use no sources/sinks. $iwells = pwells = 0$.	0 0
Card # 21	Initial concentrations.	1.0 1.0 1.0

columndata

Card # 1	$inifaceflag = 1$.	1
Card # 2	$flowflag = 0$.	0 1.0
Card # 3	None of the components are injected.	0 0 0

datarxn

Card # 1	$2 \leq odeflag \leq 4$.	3
Card # 2	$nalgebraic = 2$. Since there is no transport, equilibration is not necessary. $neqpreeq = 0$.	2 0 2
Card # 3	$eqcompId = \{ \}$.	
Card # 11	The analytic form for jacobian is supplied. Nonnegativity of concentrations is not imposed. Consistent estimates of initial derivatives are provided.	1 0 0
Card # 13	$nrxnconsts = 2$.	3
Card # 14	k_{eq1}, k_{f2} . Note that the initial conditions satisfy the algebraic relation $k_{eq1}a_0 - b_0 = 0$.	1.0 1.0

The numerical solution matches the analytic solution. For example at $t = 10.0$, the analytic solution for (a,b,c) is (0.006737947,0.006737947,2.986524106) and the numerical

Figure 4.25: Equilibrium reactions may be replaced by kinetic reactions with *sufficiently* large forward/backward rate constants.

solution (using a time-step size $\Delta t = 0.1$) is (0.006737953, 0.006737953, 2.986524094). The use of *odeflag* as 2 or 4 also yields the same solution. This problem can also be solved as an all kinetic system with large k_{f1} and k_{b1} that maintain the ratio k_{f1}/k_{b1} to be equal to $k_{eq1} = 1.0$. Figure 4.25 shows the solution for A for various values of k_{f1} ($= k_{b1}$ since $k_{eq1} = 1.0$) ranging from 1.0 to 100.0. As k_{f1} and k_{b1} grow large, the numerical solution of the all kinetic system clearly approaches that of the mixed system, as expected.

The solution for the reaction system in Equation 3.13 is shown in Figure 4.26. All the initial concentrations are equal to 1.0. The rate constants are the following. $k_{f1} = 0.1$, $k_{b1} = 1.0$, $k_{eq2} = 1.0$, $k_{f3} = 1.0$, $k_{b3} = 0.1$, and $k_{eq4} = 1.0$.

Transport + Mixed reaction system Now consider the following reaction system with transport.

Let the first reaction be under equilibrium such that,

$$k_{e1}ab - c \equiv 0.$$

Figure 4.26: Solution of the reaction system, Equation 3.13.

Following the procedure illustrated in Section 3.4 we get the following initial value DAE system.

$$\begin{Bmatrix} 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} \frac{da}{dt} \\ \frac{db}{dt} \\ \frac{dc}{dt} \\ \frac{dd}{dt} \end{Bmatrix} = \begin{Bmatrix} -k_e ab - c \\ 2k_{f2}a^2c - 2k_{b2}d \\ -a/2 + a_0/2 + 3b/2 - 3b_0/2 + c - c_0 \\ a/2 - a_0/2 - b/2 + b_0/2 + d - d_0 \end{Bmatrix}. \quad (4.6)$$

Considering the equilibrium reaction alone for equilibration and repeating the procedure yields the following nonlinear system.

$$\begin{aligned} k_e ab - c &= 0 \\ -a + a_0 + b - b_0 &= 0 \\ a - a_0 + c - c_0 &= 0 \end{aligned} \quad (4.7)$$

From Equations 4.6 and 4.7 the subroutine files `rxns.f` and `preeqrxns.f` can be coded up. It is noted that Equation 4.7 has an analytic solution that can be used to verify the code for equilibration. This analytic solution for a is the positive root of the quadratic,

$$a^2 + (b_0 + 1 - a_0)a - (a_0 + c_0)/k_e = 0.$$

b and c are then computed from Equation 4.7.

The transport problem consists of injecting component "a" into a column. Components "b", "c", and "d" are immobile. The initial concentrations of a , b and c are such that they obey the equilibrium relation $k_{e1}ab - c \equiv 0$. A few relevant cards from the files **data**, **columndata** and **datarxn** are displayed here.

data

Card # 1	We use method-1. No parameter estimation.	1 4 0
Card # 2	$ncomp = 4$.	4
Card # 3	Choose $nxp = 50$.	50 1 1
Card # 4	Choose a $\Delta t = 0.005$ and simulate till a time $t = 2.0$.	0.0 2.0 0.005 0
Card # 13	Let $L_x = L_y = L_z = 1.0$.	1.0 1.0 1.0
Card # 16	Let $k_x^h = 1.0$.	1.0 1.0 1.0
Card # 17	$\alpha_l = 0.05$ and $\alpha_t = 0.01$.	0.0 0.05 0.01
Card # 18	Use dirichlet conditions for potential in the x-direction and noflow conditions in the other two directions. Use a nonzero flux condition for concentration at outflow boundary.	1 1 0 0 0 0 2
Card # 19	Use unit potential gradient along the x-direction.	1.0 0.0 0.0 0.0 0.0 0.0 0.0
Card # 22	Initial concentrations of "a, b" and "c" satisfy the equilibrium relation.	0.4 0.5 0.2 0.0

columndata

Card # 2	Force a unit flow rate through the column.	1 1.0
Card # 3	Inject "a" into the column (at $x = 0$) and none of the other components.	1 0 0 0

Figure 4.27: Concentration profiles for components *a* and *c*, of Equation 4.5.

Card # 4 Step input of "a". 1 0 0 0

Card # 7 Injection concentration of "a" is 1.0 0.0 10.0 1.0

datarxn

Card # 1 Use DASSL. 3

Card # 2 *nalgebraic* = 3, *neqpreeq* = 3. Let us use
INTBIS for equilibration. 1 3 2

Card # 3 Components *a*, *b* and *c* participate in equi-
librium reactions. So *eqcompId* = {1 2 3}. 1 2 3

Card # 5 Use some default values. 10000 0 1.0d-7 1.0d-10

Card # 6 Search region for roots. 0.0 1.00
0.0 1.0
0.0 1.0

Figure 4.28: Concentration profiles for components b and d , of Equation 4.5.

Card # 13 $nrxnconsts = 3.$

3

Card # 14 $k_e, k_{f2}, k_{b2}.$

1.0 1.0 1.0

Figure 4.27 shows the concentration profiles in a column for component a and c at four different times; 0.5, 1.0, 1.5 and 2.0. Figure 4.28 shows the profiles for components b and d at those same times. The solution with analytic equilibration matches closely with the solution obtained by INTBIS so that they are indistinguishable in these figures.

DAE systems obtained by replacing some fast reactions with equilibrium relations could sometimes be more difficult to solve than the original stiff problem. For some problems it may be efficient to **not** to treat a very fast reaction as an equilibrium reaction (i.e. as an algebraic relation), but treat it as such (i.e. as an ODE). Figure 4.29 shows the results of column transport on the reaction system, Equation 3.4, where the reactions (2nd and 4th) are treated as either being very fast (very large k_{f2}, k_{b2}, k_{f4} , and k_{b4} in Equation 3.6) or as in equilibrium (Equations 3.5a and 3.5b). The other rate constants are the same as in Figure 4.26. $L_x = 1.0$ meters and $V_x = 1.0$ meters/day. Component "a" is injected at $x = 0.0$ at unit concentration. The initial concentrations of all components are equal to 1.0. Components "a" and "e" are mobile and the rest are immobile. Figure 4.29 shows the concentration profiles in the column for the component

Formulation	Δt (in days)	CPU time
DAE	dt = 0.01	2.75 hrs
	dt = 0.001	6.75 hrs
	dt = 0.0001	66.1 hrs
ODE	dt = 0.01	0.25 hrs
	dt = 0.001	0.5 hrs
	dt = 0.0001	3.4 hrs
	dt = 0.00005	6.0 hrs

"g", obtained by solving the DAE system with a time-step size of 0.0001 days. The solution with a bigger time-step size 0.01 days differs significantly from the solution with $\Delta t = 0.0001$ days. The reason is that at this time-step size, the solution after transport is too far out of equilibrium. When equilibrium reactions are present the time-step size needs to be small enough to not let the solution be too far away from being in equilibrium during transport.

The reaction problem is also solved as an all ODE system, by **not** assuming equilibrium for the 2nd and 4th reactions. but taking those reaction rate constants to be very large (equal to 10^8) and maintaining $\frac{k_{f2}}{k_{b2}} \equiv k_{eq2} = 1.0$ and $\frac{k_{f4}}{k_{b4}} \equiv k_{eq4} = 1.0$. A nearly converged solution is obtained with a $\Delta t = 0.00005$ days. For a given time-step size, both the ODE and DAE formulations yield nearly identical solutions. The table above shows the CPU times on a DEC ALPHA workstation, for each of these problems. Clearly for this problem, it is not efficient to replace fast reactions with algebraic relations.

Figure 4.29: Concentration profiles for component "g" in Equation 3.13.

Appendix A

Specification of Reactions

The formulation of the reaction problem for arbitrary kinetic/equilibrium reactions, the determination of the index and the generation of the appropriate subroutines to include in RAFT is automated with the help of a symbolic computational language MAPLE. The input to the MAPLE program is discussed here. It consists of specifying the number of components *ncomp*, number of reactions *nreactions* etc... For the example problem in Sections 3.4.1 and 3.4.2 the following MAPLE procedure `rxninput` is prepared.

```
#
# * This MAPLE procedure supplies the stoichiometry, reaction rate
# expressions and other information required for the formulation
# of the reaction problem and the generation of appropriate
# subroutines
#
rxninput := proc()
#
# * Declare all the GLOBAL variables that will be needed by the main
# program.
# * "component names", "reaction rate names", "rate constants",
# "equilibrium constants" etc... need to be declared.
#
global ncomp,nreactions,neqrxs,eqrxnId,rates,equilibriumeqns,
      eqcompId,eqcomp,components,ratelist,dcdt,allodes,
      c1, c2, c3, c4, c5, c6, c7, c8, r1, r2, r3, r4,
      kb1, kb2, kb3, kb4, keq2, keq4, kf1, kf2, kf3, kf4 ;
#
# * Declare the LOCAL variables that are used only in this procedure.
#
local B1, B2, B3, B4, F1, F2, F3, F4, i, dcdtarray, eqexpressarray ;
#
with(linalg) :
precision := double :
```

```

#
# * Give the total number of COMPONENTS "ncomp".
#   The components are NAMED as "c1 c2 c3 ... cncomp".
#
ncomp := 8 ;
for i from 1 to ncomp do
    components := [op(components),c.i]
od ;
#
# * Give the total number of REACTIONS "nreactions".
#   The reaction rates are NAMED as "r1 r2 r3 ... rnreact".
#
nreactions := 4 ;
for i from 1 to nreactions do
    ratelist := [op(ratelist),r.i]
od ;
#
# * Give the total number of EQUILIBRIUM reactions/expressions.
neqrxns := 2 ;
#
# * List the "indices" of these equilibrium reactions in "ratelist"
#   above.
#
eqrxnId := [2,4] ;
#
# * List the "indices" of the components in "components" above that
#   participate in equilibrium reactions.
#
eqcompId := [2,4,5,6,7,8] ;
#
# * If an ODE formulation is desired (when neqrxns := 0) then set
#   "allodes := 1". Else "allodes := 0".
#
allodes := 0 ;
#
# * Component reaction rates are "dcdtarray[1 .. ncomp]".
# * Define the rates of reaction for each component 1 ... ncomp.
#

```

```

dcdtarray[1] := -r1 - r3 ;
dcdtarray[2] := -r1 - r2 ;
dcdtarray[3] := r1 + r3 ;
dcdtarray[4] := -5*r2 + r3 ;
dcdtarray[5] := r2 - r4 ;
dcdtarray[6] := 2*r2 - r3 ;
dcdtarray[7] := -2*r4 ;
dcdtarray[8] := -2*r3 + 3*r4 ;
#
# * Give the rate expressions "r1 r2 ... r.nreact".
#   Main program assigns "r1 = rates[1] ... r.nreact = rates[nreact]".
#
F1 := kf1*c1*c2 ;
B1 := kb1*c3 ;
rates[1] := F1 - B1 ;
#
F2 := kf2*c2*(c4**5) ;
B2 := kb2*c5*c6*c6 ;
rates[2] := F2 - B2 ;
#
F3 := kf3*c1*c6*c8*c8 ;
B3 := kb3*c3*c4 ;
rates[3] := F3 - B3 ;
#
F4 := kf4*c5*c7*c7 ;
B4 := kb4*(c8**3) ;
rates[4] := F4 - B4 ;
#
# * Give the equilibrium relations ( = 0 ) for all the equilibrium
#   reactions, 1 ... neqrxns.
#
eqexpressarray[1] := keq2*c2*(c4**5) - c5*c6*c6 ;
eqexpressarray[2] := keq4*c5*c7*c7 - c8**3 ;
#
# * Make up a list of component reaction rates and equilibrium
#   expressions to be passed back to the main program.
#
for i from 1 to neqrxns do

```

```
    equilibriumeqns := [op(equilibriumeqns),eqexpressarray[i]] ;  
od ;  
for i from 1 to ncomp do  
    dcdt := [op(dcdt),dcdtarray[i]] ;  
od ;  
#  
end ;
```


Bibliography

- [1] T. Arbogast, A. Chilakapati, and M.F. Wheeler. A Characteristic-Mixed Method for Contaminant Transport and Miscible Displacement. In T.F. Russell, R.E. Ewing, C.A. Brebbia, W.G. Gray, and G.F. Pinder, editors, *Numerical Methods in Water Resources*, pages 77–84. Computational Mechanics Publications, Southampton, UK, 1992.
- [2] T. Arbogast and M.F. Wheeler. A Characteristics-Mixed Finite Element Method for Advection Dominated Transport Problems. Technical Report TR92-37, Rice University, Houston, TX, November 1992.
- [3] J. Audry-Sanchez. On the Numerical Solution of Differential Algebraic Equations. *The Canadian Journal of Chemical Engineering*, 66:1031–1035, 1988.
- [4] R.S. Bain. *NNES User's Manual, NETLIB*, March 1993.
- [5] J. Bear. *Dynamics of Fluids in Porous Media*. Dover Publications, Inc, New York, 1972.
- [6] R.C. Borden and P.B. Bedient. Transport of Dissolved Hydrocarbons Influenced by Oxygen-Limited Biodegradation 1. Theoretical Development. *Water Resources Research*, 22(13):1973–1982, 1986.
- [7] J.P. Boris and D.L. Book. Flux Corrected Transport. I: SHASTA, A Fluid Transport Algorithm that Works. *Journal of Computational Physics*, 11:38–69, 1973.
- [8] K.E. Brenan, S.L. Campbell, and L.R. Petzold. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Systems*. North-Holland, Amsterdam, 1989.
- [9] A. Chilakapati. *Numerical Simulation of Reactive Flow and Transport Through the Subsurface*. PhD thesis, Rice University, Houston, TX, 1993.
- [10] J.E. Dennis, D.M. Gay, and R.E. Welsch. An Adaptive Nonlinear Least-Squares Algorithm. *Acm Trans. Math. Software*, 7(3), 1981.
- [11] J. Dongarra, C. Moler, J. Bunch, and G. Stewart. *LINPACK Users' Guide*. SIAM, 1979.

- [12] J.C. Friedly. Extent of Reaction for Open Systems with Multiple Heterogeneous Reactions. *AIChE Journal*, 37:687-693, 1991.
- [13] J.C. Friedly and J. Rubin. Solute Transport with Multiple Equilibrium-Controlled or Kinetically Controlled Chemical Reactions. *Water Resources Research*, 28(6):1935-1953, July 1992.
- [14] A.D. Gupta, L.W. Lake, G.A. Pope, K. Sephernoori, and M.J. King. High-Resolution Monotonic Schemes for Reservoir Fluid Flow Simulation. *In Situ.*, 15(3):289-317, 1991.
- [15] E. Hairer, C. Lubich, and M. Roche. *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*. Springer-Verlag, 1989.
- [16] R.J. Hanson. Least Squares with Bounds and Linear Constraints. *SIAM J. Sci. Stat. Comput.*, 7(3):826-834, 1986.
- [17] R.G. Hills, K.A. Fisher, M.R. Kirkland, and P.J. Wierenga. Application of Flux-Corrected Transport to the Las Cruces Trench Site. *Water Resources Research*, 30(8):2377-2386, 1994.
- [18] A.C. Hindmarsh. ODEPACK, A Systemtized Collection of ODE Solvers. In R.S. Stepleman et al., editor, *Scientific Computing*, pages 55-64. North-Holland, Amsterdam, 1983.
- [19] A.C. Hindmarsh and L.R. Petzold. Algorithms and Software for Ordinary Differential Equations and Differential-Algebraic Equations, Part I: Euler Methods and Error Estimation. *Computers in Physics*, 9(1):34-41, 1995.
- [20] A.C. Hindmarsh and L.R. Petzold. Algorithms and Software for Ordinary Differential Equations and Differential-Algebraic Equations, Part II: Higher Order Methods and Software Packages. *Computers in Physics*, 9(2):148-155, 1995.
- [21] C.W. White III and W.D. Seider. Analysis of Chemical Reaction Systems. *Chem. Engg. Commun.*, 9:159-174, 1981.
- [22] R.B. Kearfott and M. Novoa III. Algorithm 681 INTBIS: A Portable Interval Newton/Bisection Package. *ACM Trans. Math. Soft.*, 16:152-157.
- [23] E.E. Kunhardt and C. Wu. Towards a More Accurate Flux-Corrected Transport Algorithm. *J.Comp.Phys*, 68:127-150, 1987.

- [24] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh. Basic Linear Algebra Subprograms for FORTRAN Usage. *Acm Trans. Math. Software*, 5:308–325, 1979.
- [25] B. Van Leer. Towards the Ultimate Conservative Difference Scheme. IV. A New Approach to Numerical Convection. *Journal of Computational Physics*, 23, 1977.
- [26] D.C. Mangold and C. Tsang. Summary of Subsurface Hydrological and Hydrochemical Models. *Reviews of Geophysics*, 29(1):51–79, February 1991.
- [27] J.J. More, B.S. Garbow, and K.E. Hillstom. *User Guide for MINPACK-1*, ANL-80-74. Argonne National Laboratory, Argonne, Illinois, 1980.
- [28] T.C. Oppe, W.D. Joubert, and D.R. Kincaid. *NSPCG User's Guide, Version 1.0*, April 1988.
- [29] L.R. Petzold. A Description of DASSL: A Differential/Algebraic System Solver. In R.S. Stepleman et al., editor, *Scientific Computing*, pages 65–68. North-Holland, Amsterdam, 1983.
- [30] J.P. Sorensen and W.E. Stewart. Structural Analysis of Multicomponent Reaction Models: Part II. Formulation of Mass Balances and Thermodynamic Constraints. *AIChE Journal*, 26(1):104–111, January 1980.
- [31] P.K. Sweby. High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws. *SIAM Journal of Numerical Analysis*, 21, 1984.
- [32] M.Th. van Genuchten. Analytical Solutions for Chemical Transport with Simultaneous Adsorption, Zero-Order Production and First-Order Decay. *Journal of Hydrology*, 49:213–233, 1981.
- [33] H.C. Yee. Construction of Explicit and Implicit Symmetric TVD Schemes and Their Applications. *Journal of Computational Physics*, 68, 1987.
- [34] G.T. Yeh and G.A. Iskra. KEMOD: A Mixed Chemical Kinetic and Equilibrium Model of Aqueous and Solid Phase Geochemical Reactions. Technical report, Pacific Northwest Laboratory, 1995.
- [35] G.T. Yeh and V.S. Tripathi. A Critical Evaluation of Recent Developments in Hydrogeochemical Transport Models of Reactive Multichemical Components. *Water Resources Research*, 25(1):93–108, January 1989.
- [36] S. Zalesak. Fully Multidimensional Flux-Corrected Transport Algorithms for Fluids. *J.Comp.Phys*, 31:335–362, 1979.

DISTRIBUTION

No. of Copies

OFFSITE

- 12 DOE/Office of Scientific and Technical
Information

T. Arbogast
Department of Computational and
Applied Mathematics
Rice University
Houston, TX 77251-1892

C. N. Dawson
Department of Computational and
Applied Mathematics
Rice University
Houston, TX 77251-1892

J. C. Friedly
Department of Chemical Engineering
University of Rochester
Rochester, NY 14627

J. Lieberman
ENSR Consulting and Engineering
35 Nagog Park
Acton, MA 01720

D. C. Mangold
Lawrence Berkeley Laboratory
University of California at Berkeley
Mail Stop 50E
Berkeley, CA 94720

C. A. Miller
Department of Chemical Engineering
Rice University
Houston, TX 77251-1892

A. J. Valocchi
Department of Civil Engineering
University of Illinois
Urbana, IL 61801-2397

G. T. Yeh
Department of Civil Engineering
Pennsylvania State University
University Park, PA 16802

No. of Copies

ONSITE

- DOE Richland Operations Office
41 Pacific Northwest Laboratory

N. J. Aimo
M. P. Bergeron
K. J. Cantrell
A. Chilakapati (10)
C. R. Cole
M. J. Fayer
M. D. Freshley
T. J. Gilmore
T. R. Ginn
D. J. Holford
C. T. Kincaid
P. J. Mellinger

Pacific Northwest Laboratory (cont'd)

F. B. Metting
P. D. Meyer
E. M. Murphy
W. E. Nichols
Y. Onishi
M. Oostrom
M. L. Rockhold
T. D. Scheibe
J. L. Smoot
C. S. Simmons
J. E. Szecsody
M. D. White
M. D. Williams
S. B. Yabusaki
J. M. Zachara
Publishing Coordination
Technical Report Files (5)