CrunchFlow

Software for Modeling Multicomponent Reactive Flow and Transport

USER'S MANUAL

Carl I. Steefel
Earth Sciences Division
Lawrence Berkeley National Laboratory
Berkeley, CA 94720 USA
CISteefel@lbl.gov

November 5, 2015

Mathematical Formulation 7
Conservation Equations 7
Example of a Total Concentration 9
Reaction Rate Laws 11
Example Involving Albite Dissolution 12
Numerical Approach 13
Global Implicit Approach 14
Operator Splitting Approach 15
Integrated Finite Difference Formulation 16
Transport Algorithm in Global Implicit Approach 17
Transport Algorithm for Operator Splitting Approach 18
Advection 18
Diffusion and Dispersion 19
Solving the Equations with Newton's Method 19
Structure of the Jacobian Matrix 20
Convergence Criteria 21
Updating Mineral Properties 22
Program Structure 23
Features of CrunchFlow 25
Units 25
Boundary Conditions 26
Choice of Primary and Secondary Species 27
Redox Reactions 28
Acid- Base Reactions 28
Activity Coefficient Model 28
Initialization Procedure 28
Temperature Gradients 29
Running CrunchFlow 30
Reading Input File Name from a File 30
Keyword Blocks 31
Space and Time Units 31
Input File Formats 32
SingleColumn Example 32
ContinuousRead Example 32
FullFormat Example 32
Unformatted Example 33
TITLE 33
RUNTIME 33
Coordinate 34
Correction_max 34
Courant_number 34
Database 35
Database_sweep 35
Debye-Huckel 35
Density_module 35
Dissolution_max 36

Fix_saturation 36
Generic_rates 36
Gimrt 37
Gimrt_pc 37
Gimrt_pclevel 37
Gimrt_solver 37
Graphics 38
Hindmarsh 38
Lag_activity 38
Later_inputfiles 38
Master 39
OvershootTolerance 39
Pc 39
Pclevel 40
Reaction_path 40
Read_SaturationFile 40
Restart 41
Save_restart 41
Screen_output 41
Solver 42
Speciate_only 42
Steady_state 42
Timestep_max 43
Timestep_init 43
Time_tolerance 43
OUTPUT 44
Spatial_profile 44
Time_series 45
Time_series_print 45
Time_series_interval 46
Time_series_output 46
MakeMovie 46
DISCRETIZATION 46
Xzones 46
Yzones 47
Zzones 47
PRIMARY_SPECIES 47
Database Format 48
Input File Entry of Primary Species 48
SECONDARY_SPECIES 49
Database Format 49
Input File Entry of Secondary Species 50
GASES 50
Database Format 51
Input File Entry of Secondary Species 51
MINERALS 51

Database Formats **51**
Thermodynamic Database Entries *51*
Kinetic Database Entries *52*
Entries in Input File **56**
AQUEOUS_KINETICS 59
Database Formats **60**
Types of Rate Laws **60**
Entries in Input File **62**
ION_EXCHANGE 62
SURFACE_COMPLEXATION 63
GEOCHEMICAL CONDITIONS 64
Units **64**
Equilibrate_surface **65**
Temperature **65**
Set_porosity **65**
Set_saturation **65**
Types of Constraints: Aqueous Species **65**
Total Concentration Constraint **66**
Species Concentration Constraint **67**
Species Activity Constraint **67**
Gas Constraint **67**
Mineral Constraint **67**
Charge Constraint **68**
Solid Phase Inputs **68**
Mineral Volume Fractions **68**
Mineral Surface Area Options **68**
Surface Hydroxyl Concentrations **70**
Exchanger Concentrations **70**
INITIAL_CONDITIONS 71
BOUNDARY_CONDITIONS 72
TRANSPORT 72
Fix_diffusion **72**
Calculate_diffusion **73**
Cementation_exponent **73**
Formation_factor **73**
Diffusion_activation **74**
D_25 .... **74**
Gas_diffusion **74**
Dispersivity **74**
Constant_tortuosity **75**
Read_TortuosityFile **75**
Tortuosity **76**
Anisotropy_ratioY **76**
Anisotropy_ratioZ **77**
Threshold_porosity **77**
Tortuosity_below **77**

Tortuosity_above **77**
FLOW **78**
Constant_flow **78**
Constant_gasflow **78**
Pump **78**
Gaspump **78**
Infiltration **79**
Read_VelocityFile **79**
Read_GasVelocityFile **80**
Calculate_flow **80**
Read_PermeabilityFile **81**
Permeability_x **82**
Permeability_y **83**
Permeability_z **83**
Gravity **83**
Pressure **84**
Initialize_hydrostatic **85**
TEMPERATURE **85**
Set_temperature **85**
Temperature_gradient **85**
Read_TemperatureFile **86**
POROSITY **86**
Fix_porosity **86**
Minimum_porosity **86**
Porosity_update **87**
Read_PorosityFile **87**
PEST **87**
CreatePestInstructionFile **87**
CreatePestExchangeFile **88**
Exchange **88**
EROSION **88**
Read_BurialFile **88**
Acknowledgements **90**
Literature Cited **90**

Mathematical Formulation

Conservation Equations

The governing equation for the conservation of solute mass is given by (Steefel, 1992; Steefel and Lasaga, 1994)

$$\frac{\partial}{\partial t}\left(\phi\rho_f M_{H_2O}C_\imath\right)+\nabla\bullet\left(-\mathbf{D}\nabla(\rho_f M_{H_2O}C_\imath)+\mathbf{u}\rho_f M_{H_2O}C_\imath\right)=R_\imath \quad (\imath=1,2,...N_{tot}), \quad (1)$$

where $C_\imath$ is the molal concentration of a species in solution (in units of moles per kg H_2O), M_{H_2O} is the mass fraction of H_2O, ρ_f is the fluid density, $\mathbf{u}$ is the Darcian flux, $\mathbf{D}$ is the combined dispersion–diffusion tensor, $R_\imath$ (in units of moles per unit volume rock per unit time) is the total reaction rate of species $\imath$ in solution, and N_{tot} is the total number of aqueous species. Note that the porosity, ϕ, is imbedded in both the flux terms and the reaction terms as described below.

The reaction term $R_\imath$ is divided into dissolution–precipitation (heterogeneous) reactions, $R_\imath^{min}$, aqueous (homogeneous) reactions, $R_\imath^{aq}$, and adsorption reactions, $R_\imath^{ads}$, such that

$$R_\imath = R_\imath^{min}+R_\imath^{aq}+R_\imath^{ads}. \quad (2)$$

The reaction rate is written here in terms of a unit volume of rock but could be written equivalently per unit volume of fluid if it were then multiplied by the porosity, ϕ. The Darcian flux is related to the true velocity of the fluid, v, by

$$\mathbf{u}=\phi\mathbf{v}. \quad (3)$$

$\mathbf{D}$ is a second order tensor which is defined as the sum of the mechanical or kinematic dispersion coefficient $\mathbf{D}^*$ and the molecular diffusion coefficient, D_0, in water divided by the formation resistivity factor, F

$$\mathbf{D}=\mathbf{D}^*+\frac{D_0}{F}. \quad (4)$$

F is defined as the ratio of the resistivity of the saturated porous medium over the resistivity of the pore solution alone (Marsily, 1986). The resistivity factor, F, may be defined in numerous ways (see, for example, Dullien, 1979), but here a definition based on Archie's Law is used which gives the formation factor as

$$F=\phi^{-m}, \quad (5)$$

where m is the “cementation exponent". Dullien (1979) reports values of m which range between about 1.3 and 2.5. The kinematic dispersion coefficient is written here as

$$D_{ii}^*=\alpha_T\,|u|+(\alpha_L-\alpha_T)\frac{u_i^2}{|u|}, \quad (6)$$

where α_L and α_T are the longitudinal and transverse dispersivities respectively (Bear, 1979; Marsily, 1986) and $|u|$ is the magnitude of the velocity. Off-diagonal dispersion coefficients (i.e., D_{ij}^*, where i and j are coordinate directions) are not included in **GIMRT** at the present time.

The heterogeneous reaction term $R_\imath^{min}$ can be written as the sum of all the individual mineral–water reactions which affect the concentration of the $\imath$ th species

$$R_\imath^{min} = -\sum_{m=1}^{N_m} \nu_{\imath m} r_m, \quad (7)$$

where r_m is the rate of precipitation or dissolution of mineral m per unit volume rock, $\nu_{\imath m}$ is the number of moles of $\imath$ in mineral m (or stoichiometric coefficient if the reaction is written in terms of the dissolution of one mole of the mineral), and N_m is the number of minerals present in the rock. Following convention, we take r_m as positive for precipitation and negative for dissolution.

Equation 1 provides a general formulation for the conservation of solute mass which makes no assumptions of chemical equilibrium. If we assume that the various aqueous species are in chemical equilibrium, however, it is possible to reduce the number of *independent* concentrations, that is, the number that actually need to be solved for. Mathematically, this means that in a system containing N_{tot} aqueous species, the number of independent chemical components in the system N_c is reduced from the total number of species by the N_x linearly independent chemical reactions between them (for further discussion, see Hooyman, 1961; Aris, 1965; Bowen, 1968; Van Zeggeren and Storey, 1970; Reed, 1982; Lichtner, 1985; Kirkner and Reeves, 1988). This leads to a natural partitioning of the system into N_c *primary* or *basis* species, designated here as $C_\jmath$, and the N_x *secondary* species, referred to as $X_\imath$ (Reed, 1982; Lichtner, 1985; Kirkner and Reeves, 1988). The equilibrium chemical reactions between the primary and secondary species take the form

$$A_\imath \rightleftharpoons \sum_{j=1}^{N_c} \nu_{\imath\jmath} A_\jmath \quad (\imath = 1,...,N_x), \qquad (8)$$

where the $A_\jmath$ and the $A_\imath$ are the chemical formulas of the primary and secondary species respectively and $\nu_{\imath\jmath}$ is the number of moles of primary species $\jmath$ in one mole of secondary species $\imath$. It should be noted here that the partitioning between the primary and secondary species is not unique, that is, we can write the chemical reactions in more than one way. The reversible reactions provide an algebraic link between the primary and secondary species via the law of mass action for each reaction

$$X_\imath = K_\imath^{-1} \gamma_\imath^{-1} \prod_{\jmath=1}^{N_c} (\gamma_\jmath C_\jmath)^{\nu_{\imath\jmath}} \quad (\imath = 1,...,N_x), \qquad (9)$$

where γ_j and γ_i are the activity coefficients for the primary and secondary species respectively, and the K_i are the equilibrium constants of the reaction given in Equation 8, written here as the destruction of one mole of the secondary species. Equation 8 implies that the rate of production of a primary component j due to homogeneous reactions can be written in terms of the sum of the total rates of production of the secondary species (Kirkner and Reeves, 1988)

$$R_j^{aq} = -\sum_{i=1}^{N_x} \nu_{ij} r_i, \quad (10)$$

where r_i are the reaction rates of the secondary species. Equation 10 suggests that one can think of a mineral dissolving, for example, as producing *only primary species* which then equilibrate instantly with the secondary species in the system. Using Equation 10, the rates of the equilibrium reactions can be eliminated (Lichtner, 1985; Kirkner and Reeves, 1988)

$$\frac{\partial}{\partial t}\left[\phi \rho_f M_{H_2O}\left(C_j + \sum_{i=1}^{N_x}\nu_{ij}X_i\right)\right] +$$

$$\nabla \bullet \left[u\rho_f M_{H_2O}\left(C_j + \sum_{i=1}^{N_x}\nu_{ij}X_i\right) - D\nabla\left\{\rho_f M_{H_2O}\left(C_j + \sum_{i=1}^{N_x}\nu_{ij}X_i\right)\right\}\right]$$

$$= R_j^{min} \quad (j = 1,...,N_c) \quad (11)$$

Note that only the term R_j^{min} remains on the right hand side of Equation 11 because we have assumed that they are the only kinetic reactions.

Example of a Total Concentration

If a total concentration, U_j, is defined (Reed, 1982; Lichtner, 1985; Kirkner and Reeves, 1988)

$$U_j = C_j + \sum_{i=1}^{N_x}\nu_{ij}X_i, \qquad (12)$$

then the governing differential equations can be written in terms of the total concentrations in the case where only aqueous (and therefore mobile) species are involved (Kirkner and Reeves, 1988)

$$\frac{\partial}{\partial t}\left(\phi\rho_f M_{H_2O}U_j\right) + \nabla \bullet \left(u\rho_f M_{H_2O}U_j - D\nabla(\rho_f M_{H_2O}U_j)\right) = R_j^{min} \quad (j = 1,...,N_c). \quad (13)$$

As pointed out by Reed (1982) and Lichtner (1985), the total concentrations can usually be interpreted in a straightforward fashion as the total elemental concentrations (e.g., total aluminum in solution), but in the case of H^+ and redox species, the total concentration has no simple physical meaning and the total concentrations may take on negative values. Such quantities, however, do appear occasionally in geochemistry, the best example of which is alkalinity. In fact, the alkalinity (which may take on negative values) is just the negative of the total H^+ concentration where $CO_2(aq)$ or H_2CO_3 is chosen as the basis species for the carbonate system. Where the system includes sorbed species (i.e., surface complexes) which are immobile, then it is a necessary to define both a mobile total concentration

$$U_j^{mobile} = C_j^{mobile} + \sum_{i=1}^{N_x} \nu_{ij} X_i^{mobile}, \qquad (14)$$

and an immobile total concentration made up of the surface complexes

$$U_j^{immobile} = C_j^{immobile} + \sum_{i=1}^{N_x} \nu_{ij} X_i^{immobile},. \quad (15)$$

The governing differential equation then becomes

$$\frac{\partial}{\partial t}\left(\phi \rho_f M_{H_2O}(U_j^{mobile} + U_j^{immobile}\right) + \qquad (16)$$
$$\nabla \bullet \left(u\rho_f M_{H_2O} U_j^{mobile} - D\nabla(\rho_f M_{H_2O} U_j^{mobile})\right) = +R_j^{min} \quad (j = 1, ..., N_c).$$

The concept of a total concentration will be clarified with a few examples. Consider a system in which the aqueous species consist of H_2O, H^+, OH^-, Al^{+++}, and $Al(OH)_4^-$. Assuming that these species are all in equilibrium, then the number of unknown concentrations (i.e., N_c) is reduced from the total number of species concentrations by the number of independent equilibrium reactions (N_x) between them

$$Al^{+++} + 4H_2O \rightleftharpoons Al(OH)_4^- + 4H^+ \quad (17)$$

$$H^+ + OH^- \rightleftharpoons H_2O \qquad (18)$$

Choosing H_2O, H^+, and Al^{+++} as the primary species and writing the chemical reactions in matrix form as in Equation 8 (that is, with the production or destruction of one mole of the secondary species), the reactions become

$$\begin{bmatrix} 4 & -4 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} H_2O \\ H^+ \\ Al^{+++} \end{bmatrix} = \begin{bmatrix} Al(OH)_4^- \\ OH^- \end{bmatrix} \qquad (19)$$

which implies that the total concentrations are

$$U_{H_2O} = C_{H_2O} + 4C_{Al(OH)_4^-} + C_{OH^-} \qquad (20)$$

$$U_{H^+} = C_{H^+} - 4C_{Al(OH)_4^-} - C_{OH^-} \quad (21)$$

$$U_{Al^{+++}} = C_{Al^{+++}} + C_{Al(OH)_4^-}. \qquad (22)$$

One can think of the total concentration, then, as a linear combination of the concentration of the primary or basis species and the amount of that primary species present in the secondary species. The U_j's, therefore, account for all of the solute mass in the system, even though in certain cases (e.g., at high pH where the concentration of OH^- is greater than the concentration of H^+), the total concentration of H^+ may be negative.

As noted above, the choice of H_2O, H^+, and Al^{+++} as the primary species is not unique and it is therefore possible to choose a different set of primary species which is mathematically equivalent. If we choose to make $Al(OH)_4^-$ the primary species, then the stoichiometric reaction matrix becomes

$$\begin{bmatrix} -4 & 4 & 1 \\ 1 & -1 & 0 \end{bmatrix} \begin{bmatrix} H_2O \\ H^+ \\ Al(OH)_4^- \end{bmatrix} = \begin{bmatrix} Al^{+++} \\ OH^- \end{bmatrix} \quad (23)$$

so that the total concentrations now become

$$U_{H_2O} = C_{H_2O} - 4C_{Al^{+++}} + C_{OH^-} \quad (24)$$

$$U_{H^+} = C_{H^+} + 4C_{Al^{+++}} - C_{OH^-} \quad (25)$$

$$U_{Al(OH)_4^-} = C_{Al(OH)_4^-} + C_{Al^{+++}} \quad (26)$$

Writing the chemical reactions differently (which mathematically is referred to as a *change of basis*), therefore, changes the total concentrations of H_2O and H^+, even though the total concentration of the Al-bearing species is the same.

Reaction Rate Laws

We use a kinetic rate law based on the assumption that attachment and detachment of ions from mineral surfaces is the rate–limiting step (i.e., a surface reaction-controlled rate law). It does not mean, however, that one cannot obtain overall transport control on the mineral dissolution or precipitation rate since this depends on the magnitude of the reaction rate relative to the macroscopic transport rates. The rate laws used for mineral precipitation and dissolution are based loosely on transition state theory (e.g., Lasaga, 1981; Aagaard and Helgeson, 1982; Lasaga, 1984). This formulation gives the dependence of the rate on the saturation state of the solution with respect to a particular mineral as a function of the ion activity product, Q_s, defined by

$$Q_s = \prod_{j=1}^{N_c} a_j^{\nu_{js}}, \quad (27)$$

where the a_j are the activities of the primary species used in writing the dissolution reaction for the mineral. In order to incorporate the strong pH dependence of most mineral dissolution and precipitation reactions far from equilibrium, parallel rate laws are used which are summed to give the overall reaction rate law for a particular mineral

$$r_s = -A_s \left\{ \sum_{l=1}^{N_{rs}} k_l \left(\prod_{i=1}^{N_c+N_x} a_i^{p_{il}^s} \right) \right\} \left[1 - \left(\frac{Q_s}{K_s} \right)^M \right]^n, \quad (28)$$

where k_l is the far from equilibrium dissolution rate constant for the l th parallel reaction, p_{il}^s gives the exponential dependence on species i of the l th parallel reaction (i.e., the reaction order), K_s is the equilibrium constant, N_{rs} is the number of parallel reactions, and A_s refers to the surface area of individual minerals in the rock matrix or fracture. The exponents n and M allow for nonlinear dependencies on the affinity term and are normally taken from experimental studies. The term $\prod_{\imath=1}^{N_c+N_x} a_\imath^p$ incorporates the effects of various ions in solution on the far from equilibrium dissolution rate. This is most commonly the solution pH or hydroxyl ion activity but may include other electrolytes as well (e.g., Na^+ and Cl^-, Dove [1995]). As an example, consider the overall rate law for kaolinite (Nagy, 1995) which includes H^+- and OH^--dependent dissolution rates

$$r_{kaol} = -A_{kaol}\left\{k_{H^+}a_{H^+}^{0.17} + k_{OH^-}a_{OH^-}^{0.54}\right\}\left[1-\frac{Q_{kaol}}{K_{kaol}}\right]. \qquad (29)$$

Note also that Equation 28 is fully compatible with the thermodynamics of the system, since if the rate constant is chosen large enough relative to the transport rates in the system, local equilibrium (or transport- controlled) behavior is obtained.

The temperature dependence of the reaction rate constant can be expressed reasonably well via an Arrhenius equation (Lasaga, 1984). Since many rate constants are reported at 25°C, it is more convenient to write the rate constant at some temperature as

$$k = k_{25}exp\left[\frac{-E_a}{R}\left(\frac{1}{T}-\frac{1}{298.15}\right)\right], \qquad (30)$$

where E_a is the activation energy, k_{25} is the rate constant at 25°C, R is the gas constant, and T is temperature in the Kelvin scale. Rate constants at temperature can be computed by providing the appropriate activation energy and the rate at 25°C, or by inputting the rate constant at temperature and specifying a zero activation energy.

Depending on how mineral surface area is computed, it may be necessary to include a dependence of the surface area on the matrix and fracture porosity such that $A_s \rightarrow 0$ as the porosity $\phi \rightarrow 0$. Mineral surface area formulations based on grain size like that presented by Lasaga (1984) cannot be used to describe the mineral surface area in contact with fluid as the porosity goes to zero (a porous medium made up of closest packed spheres cannot have a zero porosity). Here we use the expression

$$A_s(t) = A_s^\circ\left(\frac{\phi(t)}{\phi_\circ}\right), \qquad (31)$$

where A_s° and $\phi_\circ$ refer to the initial surface area and initial porosity, respectively.

Example Involving Albite Dissolution

In this section, we give an example involving albite dissolution to show how both the aqueous complexation reactions and the mineral reaction rates result in a nonlinear system. Here we consider only the ODEs which result from the reaction term (i.e., the equations would be those in the reaction module when using either operator splitting or sequential iteration). If a global implicit approach were used, one would also have terms involving spatial derivatives in the equation. If we consider a geochemical system involving the primary species H_2O, H^+, Al^{+3}, $SiO_{2(aq)}$, and Na^+ and the secondary species (all in equilibrium with the primary species) OH^-, $AlOH_4^-$, $H_3SiO_4^-$, and $NaCl_{(aq)}$, then the differential equations take the form

$$\frac{\mathrm{d}(Na^+ + NaCl_{(aq)})}{\mathrm{d}t} = A_{alb}\, a_{H^+}^{0.4}\, k_{alb} \left[1 - \left(\frac{a_{Na^+} a_{Al^{+3}} a_{Sio_{2(aq)}}^3}{a_{H^+}^4} \right) K_{alb}^{-1} \right],$$

$$\frac{\mathrm{d}(Sio_{2(aq)} + H_3SiO_4^-)}{\mathrm{d}t} = 3\, A_{alb}\, a_{H^+}^{0.4}\, k_{alb} \left[1 - \left(\frac{a_{Na^+} a_{Al^{+3}} a_{Sio_{2(aq)}}^3}{a_{H^+}^4} \right) K_{alb}^{-1} \right],$$

$$\frac{\mathrm{d}(Al^{+3} + AlOH_4^-)}{\mathrm{d}t} = A_{alb}\, a_{H^+}^{0.4}\, k_{alb} \left[1 - \left(\frac{a_{Na^+} a_{Al^{+3}} a_{Sio_{2(aq)}}^3}{a_{H^+}^4} \right) K_{alb}^{-1} \right],$$

$$\frac{\mathrm{d}(H^+ - OH^- - 4AlOH_4^- - H_3SiO_4^-)}{\mathrm{d}t} = -4\, A_{alb}\, a_{H^+}^{0.4}\, k_{alb} \left[1 - \left(\frac{a_{Na^+} a_{Al^{+3}} a_{Sio_{2(aq)}}^3}{a_{H^+}^4} \right) K_{alb}^{-1} \right],$$

where we have neglected the mass balance of H_2O. The actual form of the equations solved by **GIMRT** eliminates the secondary species entirely, writing each in terms of the primary species using Equation 9.

Numerical Approach

GIMRT uses a *global implicit* or *one–step* method to solve the reaction–transport equation. In contrast, **OS3D** uses either classic time splitting of the transport and reaction terms or it can be run with the *sequential iteration* method proposed by Yeh and Tripathi (1989). There are some advantages to each approach. Yeh and Tripathi (1989) cite the major advantage of the operator splitting or sequential iteration approach as the lower memory requirements of these methods compared to the global implicit and the greater speed with which a single time step can be completed. Although these may be advantages of the time splitting approach over the global implicit, probably the most significant advantage of the time splitting approach is the ability to use algorithms for high Peclet number transport (i.e., advection dominating over dispersion and molecular diffusion) which have less numerical dispersion than those readily usable in a global implicit scheme. **GIMRT**, for example, using a first–order accurate upwind scheme, which along with the backwards differentiation, leads to a significant amount of numerical dispersion. In contrast, **OS3D** uses a third order accurate total variation diminishing or TVD method proposed by Gupta et al. (1991). The TVD algorithm results in signficantly less numerical dispersion than the upwind scheme used in the global implicit.

The chief advantage of the global implicit approach, is that unlike the TVD method, one is not restricted rigorously to the Courant condition, which requires that mass not be transported more than a single grid cell in any one timestep. Although this is an important criterion to observe in order to maintain accuracy in transient problems, it is less important once the aqueous concentrations approach a quasi–stationary state. In simulations we have carried out, we find that the operator splitting error is minimal when the Courant number is less than about 0.2, so even if one does not use an explicit transport step, minimizing operator splitting error requires a small timestep. If the interest in the problem is to model water-rock interaction over geological periods of time, then transport errors associated with the transient propagation of concentration fronts are less significant. In these cases, the global implicit approach can offer a real advantage because of the ability to take larger time steps once the system relaxes to a quasi–stationary state. In contrast, stability requirements dictate that the TVD algorithm, which uses an explicit or forward Euler method, use a timestep smaller than a Courant number of approximately 1/2.

Global Implicit Approach

In the *one–step* method employed by **GIMRT**, the transport and reaction terms are solved simultaneously. One approach would be to solve directly for the total concentrations (the U_j's) in any one iteration and then follow this with a distribution of species calculation to determine the concentrations of the individual species (see discussion by Kirkner and Reeves, 1988). The method implemented here, however, consists of solving simultaneously for the complexation (which are assumed to be at equilibrium), the heterogeneous reactions, and transport terms. This means that the primary species (the C_j's) rather than the total concentrations are the unknowns. Following the notation of Lichtner (1992), we can introduce the differential operator

$$L(U_j) = \left[\frac{\partial}{\partial t}\phi + \nabla \bullet \left(\mathbf{u} - \mathbf{D}\nabla \right) \right] U_j, \quad (32)$$

if we drop the density and mass fraction of H_2O terms for simplicity and write the governing differential equations in terms of the total component concentrations as

$$L(U_j) = R_j^{min} \quad (j = 1, ..., N_c), \quad (33)$$

where the reaction term R_j^{min} includes only irreversible (in our case, heterogeneous) reactions. Note that in this formulation, we assume that the diffusion coefficients are the same for all of the aqueous species (both primary and secondary). This makes the solution of the reaction–transport equation much simpler because the secondary species do not have to be individually transported.

We proceed by substituting Equation 9 into Equation 11 to obtain an expression for the total soluble concentration in terms of the primary species alone. Substituting this result into Equation 33 gives (essentially a kinetic version of the formulation A of Kirkner and Reeves [1988])

$$L\left[C_j + \sum_{i=1}^{N_x} \nu_{ij} \gamma_i^{-1} K_i^{-1} \prod_{j=1}^{N_c} (\gamma_j C_j)^{\nu_{ij}} \right] +$$

$$\sum_{m=1}^{N_m} \nu_{im} sgn\left(\log[Q_/ K_m] \right) A_m k_m \left| \left(\prod_{j=1}^{N_c} (\gamma_j C_j)^{\nu_{mj}} K_m^{-1} \right)^M - 1 \right|^n = 0$$

$$j = 1, \cdots N_c, (34)$$

where the γ_i's and γ_j's are the activity coefficients for the secondary and primary species respectively. The integrated finite difference method is used to convert the differential operator L into an algebraic expression and is discussed in the next section.

Operator Splitting Approach

Unlike **GIMRT**, the code **OS3D** solves the transport terms and the reaction terms separately. Splitting of the transport and reaction steps is presently carried out in **OS3D** using the Strang scheme, in which a half transport step is followed by a full reaction step which is in turn followed by another half transport step (Strang, 1968; Zysset et al., 1994). This form of time splitting involves a three-step algorithm which requires no iteration and takes the form

$$\frac{(C_k^{pre-react} - C_k^n)}{\Delta t/2} = L(C_k)^n, \quad (k = 1, ..., N_c + N_x), \quad (33)$$

followed by a reaction step

$$\frac{(U_j^{reacted}(c_j) - U_j^{pre-react}(c_j))}{\Delta t} = R_j^{min} \quad (j = 1, ..., N_c), \qquad (34)$$

which is in turn followed by another 1/2 transport step

$$\frac{(C_k^{n+1} - C_k^{reacted})}{\Delta t/2} = L(C_k)^{reacted}, \quad (k = 1, ..., N_c + N_x). \qquad (35)$$

In this scheme, $C_k^{pre-react}$ are the concentrations of the individual species at the end of the first 1/2 transport step, $U_k^{reacted}$ are the *total* concentrations at the end of the full reaction step, and C_k^{n+1} are the individual species concentrations at the end of the second 1/2 transport step (i.e., at the end of the full reaction-transport time step). L refers to the spatial differential operator defined by

$$L = \left[\nabla \bullet \left(\mathbf{u} - \mathbf{D}\nabla\right)\right]. \qquad (36)$$

This scheme is reported to be second order accurate in time (Zysset et al., 1994).

The form of the equation solved in the reaction step is the same as in Equation 32, except that in the operator splitting case we do not have the spatial operator, i.e., the operator becomes

$$L(U_j) = \left[\frac{\partial}{\partial t}\phi\right] U_j. \qquad (37)$$

Yeh and Tripathi (1989) and Valocchi and Malmstead (1989) pointed out that in some cases the classic time splitting approach described above can lead to operator splitting error. The global implicit approach, of course, eliminates this error, but at the cost of adding in transport errors associated with the spatial discretization. An alternative is to iterate sequentially between the transport and reaction in order to obtain a solution free from operator splitting error. The sequential iteration method offers a way of doing this while still avoiding the construction and solution of a large global matrix as in the global implicit approach. Our experience, however, is that the sequential iteration scheme offers little improvement over the Strang operating splitting procedure and that it involves considerably more CPU time. *At the present time, therefore, we* *recommend using the Strang operator splitting scheme rather than sequential iteration.* We have incorporated the sequential iteration approach nontheless and describe it below.

Integrated Finite Difference Formulation

The set of partial differential equations represented by Equation 34 is discretized using the *integrated finite difference* method (Patankar, 1980; Marsily, 1986). As an example of how the discretization proceeds, consider a set of partial differential equations for the total soluble concentration where the only important transport is via diffusion in one dimension and for simplicity we restrict ourselves to steady state conditions and to a system with constant density. The differential equation is then

$$\frac{\partial}{\partial x}\left(\phi D \frac{\partial U_j}{\partial x}\right) - R_j^{min} = 0. \qquad (35)$$

In the integrated finite difference formulation, a control volume is defined and the differential equations are converted to a set of algebraic equations which include the fluxes across the boundaries of the volume and a source or reaction term. Equation 35 can be written (Patankar, 1980)

$$\left(D\frac{\partial U_j}{\partial x}\right)_e - \left(D\frac{\partial U_j}{\partial x}\right)_w - \int_w^e R_j^{min} dx = 0, \qquad (36)$$

where the subscripts e and w are the derivatives evaluated at the two boundaries of the control volume. Equation 36 can be expressed as a set of algebraic equations using piecewise–linear representations of the derivatives (Patankar, 1980)

$$\frac{D_e(u_j^E - u_j^P)}{(dx)_e} - \frac{D_w(u_j^P - u_j^W)}{(dx)_w} - R_j^{min}\Delta x = 0, (37)$$

where D_e and D_w refer to the appropriate value for the diffusion coefficients at the boundary of the volume (averaged between the two grid points) and the U_j's superscripts (E, W, and P) are the total concentrations evaluated at the nodal points themselves (Figure 1). The term R_j^{min} is a function of the primary species at the nodal point P.

Following the terminology of Patankar (1980), the discretized equations can be recast in the form

$$a_P U_j^P - a_E U_j^E - a_W U_j^W - d = 0, \qquad (38)$$

where

$$a_E = \frac{D_e}{(dx)_e} \; (39)$$

$$a_W = \frac{D_w}{(dx)_w} (40)$$

$$a_P = a_E + a_W \qquad (41)$$

$$d = R_j^{min}\Delta x. (42)$$

Transport Algorithm in Global Implicit Approach

The addition of an advection term makes the set of partial differential equations much more difficult to solve stably and accurately (e.g., Press et al., 1986; Patel et al., 1988), although the set of equations can still be cast in the general form of Equation 38. One way to represent the first derivative which appears in the advection term is to use a central difference method

$$\left(\frac{\partial U}{\partial x}\right)_P = \frac{U_E - U_W}{2\Delta x} \qquad (43)$$

where the subscript P indicates that the derivative is evaluated at the nodal point P and Δx is 1/2($\delta x_e + \delta x_w$). The truncation error for the central difference representation is said to be of the order Δx^2 because in the Taylor series expansion upon which it is based, only quadratic terms and higher are neglected (Lapidus and Pinder, 1982). In contrast, a forward difference representation, given by

$$\left(\frac{\partial U}{\partial x}\right)_P = \frac{U_E - U_P}{(\delta x)_e}, \qquad (44)$$

has a truncation error of $O(x)$, that is, all terms in the Taylor series higher than first order are truncated. Despite the smaller truncation error of the central difference method, the method may not be numerically stable for advection–dominant problems because of the non–physical oscillations which can occur in the vicinity of sharp concentration fronts (Patankar, 1980; Daus and Frind, 1985; Frind and Germain, 1986; Patel et al, 1988).

It appears, therefore, that the numerical stability of a particular finite difference formulation for the first derivative depends on the relative magnitudes of the advection and dispersion/diffusion terms. The relative importance of the two can be described by the dimensionless *grid Peclet number*

$$Pe_{grid} = \frac{v\Delta x}{D} \qquad (45)$$

where Δx refers to the grid spacing at any particular point in space. Only for $Pe_{grid} < 2$ is the central difference formulation unconditionally stable (Daus and Frind, 1985; Frind and Germain, 1986; Patel et al, 1988). The problem of stability can be “cured" by using a forward difference formulation written in terms of the grid point itself (P) and the *upstream* or *upwind* node (Press et al., 1986). It should be pointed out, however, that the upstream–weighted formulation, in addition to having a larger truncation error than the central difference method, achieves its greater stability essentially by adding numerical dispersion (i.e., over and above the physical dispersion present in the problem). In a one–dimensional problem, however, numerical dispersion presents a problem only in tracking transient concentration fronts. The effect does not appear in steady state one–dimensional systems (Patankar, 1980). Numerical dispersion can be reduced by refining the grid in the vicinity of a sharp concentration front.

To take advantage of the greater accuracy of the central difference method at small values of Pe_{grid}, a *power law scheme* proposed by Patankar (1980) is used. This method slides between a fully centered form at $Pe_{grid} < 2$ to an upstream–weighted formulation at $Pe_{grid} > 10$, where the upwind scheme is necessary for stability. Using the power–law scheme of Patankar (1980), the finite difference coefficients become

$$a_E = D_e \left\| 0, \left(1 - \frac{0.1 \delta x_e v_e}{D_e} \right)^5 \right\| + \left\| 0, -v_e \right\|, \qquad (46)$$

$$a_W = D_w \left\| 0, \left(1 - \frac{0.1 \delta x_w v_w}{D_w} \right)^5 \right\| + \left\| 0, v_w \right\|, \qquad (47)$$

$$a_P = a_E + a_W, \qquad (48)$$

where the operator $\|A, B\|$ means that the greater of the values A or B is used (equivalent to the FORTRAN operator **AMAX1(A,B)**) and $|A|$ refers to the absolute magnitude of the value. A fuller description may be found in Patankar (1980).

A fully implicit backward method is used to represent the time derivative (Lapidus and Pinder, 1982). The equations to be solved for each component at each grid point at the $(t + \Delta t)$ time level are then

$$\frac{\Delta x}{\Delta t}\left(U_J^P(t+\Delta t) - U_J^P(t)\right) + a_P U_J^P(t+\Delta t) - a_E U_J^E(t+\Delta t) - a_W U_J^W(t+\Delta t) - R_J^{min}(t+\Delta t)\Delta x = 0, \quad (49)$$

where Δt is the time step and (t) refers to the values at the present time level. As discussed above, the equations are written in terms of the concentrations of the primary species in the code **GIMRT** (see Equation 34) and are expressed in terms of the total concentrations here in order to be concise.

Transport Algorithm for Operator Splitting Approach

Advection

When solutions require accurate resolution of concentration fronts in advection-dominant problems, high-resolution shock-capturing techniques are often employed. These solution techniques are typically based on explicit numerical schemes that are incompatible with a globally implicit solution of coupled transport and reactions (e.g., **GIMRT**). Decoupling (i.e., time splitting) the transport algorithm from the geochemical reaction algorithm relaxes the requirement for an implicit transport method, allowing the use of high-resolution shock-capturing techniques based on explicit numerical schemes. The principal trade-off for the additional accuracy is a strict limit on the ratio of time-step to grid spacing, a limit that is inherent in explicit transport methods. Typically, this limit becomes most restrictive under steady-state or near steady-state conditions. (Note that steady-state does not mean static processes. In physical systems, steady-state occurs when dynamic processes are balanced or fully compensating.)

In OS3D, an explicit, third-order, total variation diminishing (TVD) numerical scheme (Gupta et al., 1991) is used to model non-dispersive advective transport. The technique is based on calculating an estimated concentration at the face of a grid cell that is dependent on the flow direction. In the case of flow in the positive x-direction (i.e., $u > 0$, the estimated concentration would be:

$$C_{i+1/2}^{n+1/2} = C_i^n + \beta(r)\left(\frac{C_{i+1}^n - C_i^n}{2}\right), \qquad (56)$$

where

$$\beta(r) = \mathrm{Max}\left(0, \mathrm{Min}\left[2, 2r, \frac{2+r}{3}\right]\right), \ (57)$$

and

$$r = \left[\frac{C_i^n - C_{i-1}^n}{\delta x_i + \delta x_{i-1}}\right]\left[\frac{C_{i+1}^n - C_i^n}{\delta x_i + \delta x_{i+1}}\right]^{-1}. \qquad (58)$$

Similar calculations are performed for negative velocities and the method is generalized for multiple dimensions. The technique is third-order accurate in smooth regions, oscillation-free along concentration fronts, and does not exhibit overcompression of fronts in the presence of physical dispersion.

Diffusion and Dispersion

After the advection calculation, OS3D proceeds to solve for diffusion and dispersion. OS3D assumes that the principal direction of flow is aligned with the grid (i.e., D is a diagonal tensor). A centered numerical scheme, second-order accurate in space and time, is applied to the diffusion equation. The diffusion/dispersion equations are solved with the **WATSOLV** sparse matrix solver of VanderKwaak et al. (1995).

The user should understand that parameterizations of hydrodynamic dispersion attempt to account for advection resulting from variations in the velocity field that cannot be resolved by the numerical grid. In this standard representation based on FickÕs law, dispersion is driven by the gradient of concentration and parameterized by a linear proportionality constant (i.e., the diffusion coefficient). This is a gross simplification of a complex and poorly understood phenomenon. As such, the formulation neglects many features that have been observed in real systems. Principally, the user should be aware that unlike molecular diffusion, dispersion is length and time scale- dependent.because of the inhomogeneity of the physical media. As plumes start to spread, they are exposed to larger and larger scales of spatially variable material properties. Consequently, it is impossible to define dispersion parameters that are effective for all time. Moreover, the magnitude of the dispersion parameter is dependent upon the grid resolution: larger for large grid cells and smaller for small grid cells.

Solving the Equations with Newton’s Method

As an example, consider a problem involving one–dimensional multicomponent reactive transport. Equation 49 for any component j at the grid point P is a nonlinear function of the concentrations of the primary species at the P, E, and W nodal points using the global implicit approach of **GIMRT**. An iterative scheme, therefore, is required to solve the set of nonlinear equations in either case. The code solves the nonlinear equations with Newton’s method which makes use of a Taylor series expansion to linearize the equations. Using Newton’s method, the linearized set of equations for the 1D global implicit case become

$$\sum_{j'=1}^{N_c} \frac{\partial f_j^P}{\partial C_{j'}^P} \delta C_{j'}^P + \sum_{j'=1}^{N_c} \frac{\partial f_j^P}{\partial C_{j'}^E} \delta C_{j'}^E + \sum_{j'=1}^{N_c} \frac{\partial f_j^P}{\partial C_{j'}^W} \delta C_{j'}^W = -f_j^P, \quad (50)$$

where f_j is the discretized differential equation for the total concentration of each of the primary species. In a 2D problem, one may have dependencies, of course, on other neighboring grid cells in the Y direction (i.e., north and south of the grid cell of interest).

The individual terms, $\partial f_j / \partial C_{j'}$, make up the elements of the *Jacobian* submatrices. From Equation 49, it follows that the Jacobian submatrices can be written further as

$$\frac{\partial f_j^P}{\partial C_{j'}^P} = \frac{\partial U_j^P}{\partial C_{j'}^P}\left(\frac{\Delta x}{\Delta t} + a^P\right) - \frac{\partial R_j^{min}}{\partial C_{j'}^P}\Delta x, \quad (51)$$

$$\frac{\partial f_j^P}{\partial C_{j'}^E} = -a^E \frac{\partial U_j^E}{\partial C_{j'}^E}, \qquad (52)$$

and

$$\frac{\partial f_j^P}{\partial C_{j'}^W} = -a^W \frac{\partial U_j^W}{\partial C_{j'}^W}. \qquad (53)$$

If we define f_j as making up the elements of the vector **b** when taken over all of the grid points in the system and the $\partial f_j / \partial C_{j'}$ as forming the elements of the global Jacobian matrix **A**, then Equation 50 can be written in the familiar form

$$\mathbf{A} \cdot \delta\mathbf{C} = \mathbf{b}. \ (54)$$

Once the $\delta C_{j'}$ are computed, they are used to update the concentrations of the primary species

$$C_{j'}^{new} = C_{j'}^{old} + \delta C_{j'} \quad (j' = 1, \ldots, N_c \times M). \quad (55)$$

This completes a single Newton iteration. The procedure is repeated until convergence of the function residuals (the f_j's) are reduced to a tolerance set in the code, i.e.,

$$f_j \approx 0. \qquad (56)$$

Structure of the Jacobian Matrix

It is instructive to examine the structure of the Jacobian matrix which arises from the discretization of the coupled reaction–transport equations, since it has a bearing on the choice of a method to solve the set of linear equations given by Equation 34. The Jacobian matrices exhibit a *block tridiagonal* structure in a one–dimensional system. The blocks making up the global matrix **A** are N_c by N_c submatrices corresponding to the primary species at any grid point. That is, if we are writing the conservation equation for $SiO_{2,aq}$ at a particular point in space, the equation $f_{SiO_{2,aq}}$ will be differentiated with respect to all N_c primary species (e.g., $SiO_{2,aq}$, Al^{+++}, K^+, etc.). As an example, consider a one–dimensional system described by 4 primary species which is discretized using 4 grid points (this is for illustrative purposes only, since hopefully one would not attempt to solve differential equations using only 4 grid points). The matrix $\mathbf{A}$ can be represented compactly as a series of 4 $\times$ 4 submatrices, $\mathbf{A}_{JJ'}$, such that

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} & 0 & 0 \\ \mathbf{A}_{21} & \mathbf{A}_{22} & \mathbf{A}_{23} & 0 \\ 0 & \mathbf{A}_{32} & \mathbf{A}_{33} & \mathbf{A}_{34} \\ 0 & 0 & \mathbf{A}_{43} & \mathbf{A}_{44} \end{bmatrix}. \quad (57)$$

Those readers with some familiarity with numerical methods will recognize that this is the structure of any finite difference formulation of a partial differential equation (e.g., the temperature equation), except that in this case the entries are submatrices rather than single coefficients. If we reduced the system to a single chemical component, then the forms of the temperature and reaction–transport matrices would be identical. The submatrices need not be filled entirely with non–zero elements, but, in general, they will be dense in systems with either a large number of minerals or a great deal of complexation. The diagonal submatrices ($\mathbf{A}_{11}$, $\mathbf{A}_{22}$ etc.) contain contributions from both the heterogeneous reaction term, R_J^{min}, and from the total soluble concentrations (the U_J's). The offdiagonal submatrices, in contrast, contain only contributions from the total soluble concentrations at the neighboring grid points. Note that the global matrix $\mathbf{A}$ is sparse (i.e., the ratio of zero to non–zero entries in the matrix is large) when the number of grid points is large (e.g., greater than 100), even though the submatrices (reflecting the coupling between the various species due to reactions) may themselves be fairly dense.

Any method used to solve the set of equations described by Equation 50 which arise in the case of the global implicit method should take into account both the sparseness of the Jacobian matrix $\mathbf{A}$.

Convergence Criteria

Convergence is obtained where all of the function residuals have been reduced below a tolerance set by the parameters `atol` and `rtol` in the driver routine `gimrt.f`. These parameters are located in data statements at the top of the named routines. The tolerance on function residuals is computed from

$$tolmax = atol + rtol\left(C_J\right), \qquad (58)$$

where C_J refers to the primary species concentrations. If they are changed, the code should be recompiled. The functions residuals, when multiplied by the timestep `delt` in order to compare to the tolerances `atol` and `rtol`, have units of moles per m^3 bulk volume. The default setting for the absolute tolerance is 10^{-9} mole m^{-3} which corresponds to 10^{-12} mole liter^{-1}. Convergence will also be obtained where either the absolute value of the log concentration correction or the linear concentration correction drops below 10^{-15} (i.e., approaching the roundoff error of the computer).

Updating Mineral Properties

When convergence is achieved, the mineral volumes and surface areas (and porosity, if desired) are updated. The individual mineral volume fractions, ϕ_m, are updated with the expression

$$\phi_m(t+\Delta t) = \phi_m(t) + V_m r_m(t+\Delta t)\Delta t, \quad (59)$$

where V_m is molar volume of the mineral and r_m is its reaction rate. There are now threed different options for treating porosity (described fully in the section on the input file). One option is to specify a single value for the porosity which then remains constant for the duration of the simulation. This is set at the top of the input file using the parameters `jpor` and `constantpor`. In this case, the value of `constantpor` overrides any choice of the mineral volume fractions below. The second option is to have the porosity computed from the sum of the mineral volume fractions which may vary from grid cell to grid cell, but to then leave the porosity constant through the course of the simulation. The third option is to compute the porosity locally from the sum of the mineral volume fractions and to update it as a result of mineral precipitation and dissolution.

Where the mineral volume fractions are used, the porosity is computed from the

$$\phi = 1 - \sum_{k=1}^{N_m} \phi_m. \qquad (60)$$

Note that in this formulation, both mineral volume fractions and reactive surfaces are assumed constant for any one time step. This decoupled, two step approach normally works well since the volume fractions of the minerals usually evolve much more slowly than the solute concentrations. However, in certain cases, the approach may lead to inaccurate solutions. In previous versions, this was dealt with in two ways: 1) reducing the time step where the mineral volume fraction would go below zero and 2) limiting the amount of dissolution and/or precipitation that can occur in any one time step. Currently the second method is still used, but now the code automatically checks to see if the computed reaction rate exceeds the available mineral within the elemental volume. If it does, it calculates the reaction rate needed to cause the mineral to disappear and uses this as the reaction rate rather than the initially calculated one. In this way, there is no problem of a mass balance error (i.e., dissolving more mineral than is actually present). The approach, since it involves moving the reaction term to the right hand side of the equations as a source term, sometimes leads to temporary convergence problems, but these have not been particularly severe so far. In addition, one can specify a maximum rate of increase (using the parameter `vpptmax` described below) or decrease (using the parameter `vdissmax` described below) of the mineral volume fractions. The two step approach employed here may require smaller time steps when either the dissolving or precipitating phase has a high solubility or where very little of it is present. Where only small amounts of a mineral are present, than it is possible that dissolution of the entire amount present within a grid volume may occur in a single time step. This typically results in either inaccurate propagation of a mineral front or in oscillations in the mineral volume fraction profile.

Initial reactive surface areas are specified along with initial mineral volume fractions by the user at startup. Reactive surface areas of minerals are updated differently at the present time depending on whether the minerals are primary or secondary. For primary minerals, the surface areas are given by

$$A_m = A_{m,0} \begin{Bmatrix} \left[\left(\dfrac{\phi}{\phi_0} \right) \left(\dfrac{\phi_m}{\phi_{m,0}} \right) \right]^{2/3} & \textit{dissolution} \\ \left[\dfrac{\phi}{\phi_0} \right]^{2/3} & \textit{precipitation} \end{Bmatrix}, \quad (61)$$

where $\phi_{m,0}$ is the initial volume fraction of the mineral m and ϕ_0 is the initial porosity of the medium. This formulation ensures that as the volume fraction, ϕ_m, of a mineral $\rightarrow$ 0, its surface area also $\rightarrow$ 0. In addition, for both dissolving and precipitating minerals the term $\left(\phi/\phi_0\right)^{2/3}$ requires that the surface area of a mineral in contact with fluid $\rightarrow$ 0 when the porosity of the medium $\rightarrow$ 0. For secondary minerals (i.e., those initially with a volume fraction = 0), the reactive surface areas as a function of time are given by

$$A_m = A_{m,0} \begin{Bmatrix} \left[\left(\dfrac{\phi}{\phi_0} \right) \phi_m \right]^{2/3} & \textit{dissolution} \\ \left[\dfrac{\phi}{\phi_0} \right]^{2/3} & \textit{precipitation} \end{Bmatrix}, \quad (62)$$

which implies that if a secondary were to begin to dissolve, its surface area would equal its “initial" surface area, $A_{m,0}$, when V_m = 1 (if the porosity were constant).

Program Structure

The basic structure of the global implicit approach (**GIMRT**) is summarized in Figure 2. After specifying the initial and boundary conditions, the code begins stepping through time. Within each time step, a series of Newton iterations is required because of the nonlinearity of the equations. Each Newton iteration consists of calculating the function residuals (the f_J’s given by Equation 55) and the partial derivatives of these functions (the Jacobian elements) with respect to the unknown concentrations (Equation 59) and then assembling and solving the linear set of $N_c \times M$ algebraic equations in order to obtain the corrections to the component concentrations.

Unlike the operator splitting approach in **OS3D**, the residual functions and the Jacobian matrix are computed over the entire spatial domain before solving a global system of linearized equations. Once convergence is achieved, the mineral volume fractions and mineral surface areas are updated. If the reaction–transport calculations are combined with a calculation of the flow and temperature field, these would normally be updated as well at this time. At the end of each time step, an algorithm which computes the second derivative of the solute concentrations with respect to time is used to determine whether the time step should be increased or decreased. This algorithm provides a way of minimizing the time truncation error (if that is an issue in a particular problem) and also of controlling the size of the time step so as to maintain numerical stability.

The program structure for the time splitting (operator splitting and sequential iteration) method is similar, differing primarily in the fact that the transport is carried out in advance of any calculations of the reactions (Figure 3). The reaction calculations are then calculated individually at each point in space. Since these reactions are not directly coupled via transport to the reactions in neighboring cells, the calculations can be carried out independently. The size of the system of equations to be solved at each grid point is then N $_c$ by N $_c$ and there is no need to assemble the global system of function residuals and the global Jacobian matrix over the entire spatial domain. The sequential iteration method differs in that 1/2 of the reaction rate is included as a source term in advance of the "reaction module" and so iteration between transport and reaction is required to achieve convergence (Figure 3). Mineral volume fractions and the porosity are updated *after* all of the species have been reacted at all the space points, in a similar fashion to the global implicit.

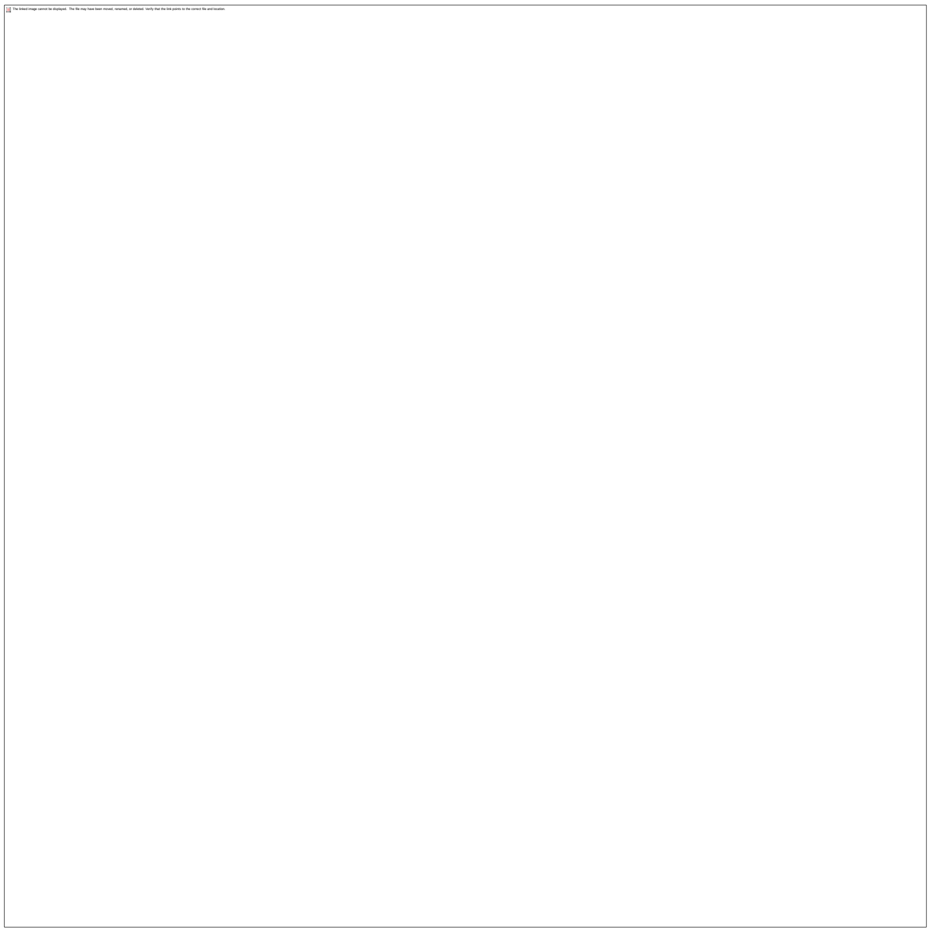
The linked image cannot be displayed. The file may have been moved, renamed, or deleted. Verify that the link points to the correct file and location.

Figure 2. Program structure for GIMRT which uses a global implicit or one-step method used to solve the multi–component reaction-transport equation.

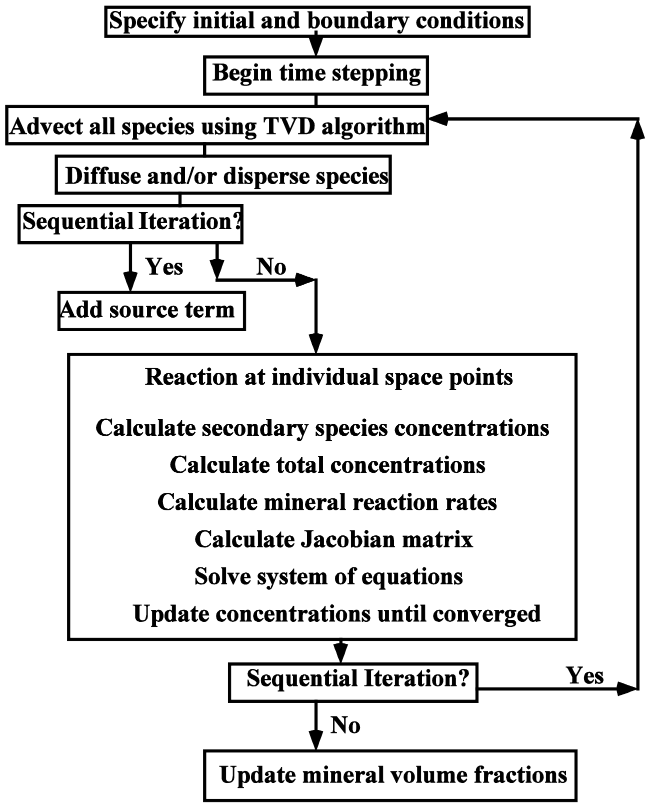
Specify initial and boundary conditions
Begin time stepping
Advect all species using TVD algorithm
Diffuse and/or disperse species
Sequential Iteration?
Yes
No
Add source term
Reaction at individual space points
Calculate secondary species concentrations
Calculate total concentrations
Calculate mineral reaction rates
Calculate Jacobian matrix
Solve system of equations
Update concentrations until converged
Sequential Iteration?
Yes
No
Update mineral volume fractions

Figure 3. Program structure for OS3D which uses time splitting of the transport and reaction step or a sequential iteration between transport and reaction.

Features of CrunchFlow

Before running the code, the user of **GIMRT** needs to decide what physical process(es) he or she wants to model and design the simulation accordingly. This involves choosing the boundary conditions for the problem (e.g., no flux boundaries or fixed concentration boundaries) and setting up the initial grid and/or initial mineral zones. The user must decide how many grid points are needed to solve a particular problem. Typically one proceeds by carrying out the simulations on a relatively coarse grid (e.g., 30 to 50 grid points) and only refining the grid later when one has a general idea of the behavior of the system. The user also needs to decide what level of chemical complexity they wish to consider.

This can involve choices in the number of independent components (primary species), the number of secondary species, and the number of reacting minerals. The discussion in the sections which follow is designed to make the basis for these decisions clearer. These sections are then followed by a step by step discussion of the input file which is needed to run **GIMRT**.

Units

Although there are some advantages to allowing the input of data in any self-consistent set of units, many of the parameters used in **GIMRT** are given commonly in certain units, so we have decided to require these units. All units for input parameters (e.g., velocities, reaction rates etc.) are given in the description of the input file. All distances are in meters. The code assumes that the reaction rates for minerals are in units of mol m^{-2} s^{-1} (these are converted in the code to mol m^{-2} yr^{-1}. Diffusion coefficients are in units of m^{-2} s^{-1}. All other references to time (time for plot files, timesteps etc.) are in years. Velocities, for example, are in units of m yr^{-1}. Concentrations for aqueous species are in units of mol kg^{-1} water (i.e., molality). The partial pressure of gases are specified in bars.

Boundary Conditions

At the present time there are three possible boundary conditions which can be used in **GIMRT** and **OS3D**. The most straightforward way to describe the possible boundary conditions in the codes is to actually begin with the *third* type of boundary (also sometimes called a Danckwerts boundary conditon). This is a boundary condition which requires continuity of flux across the boundary. Both codes assume that the flux at these boundaries is purely advective (i.e., no diffusion or dispersion occurs upstream). Where the flow is into the system, the flux at the first interior nodes downstream is assumed to be just uC_{ext}, the same as the advective flux at the upstream, exterior node. In this special case where the dispersive and diffusive flux are neglected, the *third* type of boundary condition is the same as a Dirichlet or fixed concentration boundary condition. At the boundary nodes where the flow passes out of the domain the flux is also assumed to be purely advective, so that it passes out of the system with the downstream, exterior node having no effect on the system. Note that when the flow across a boundary is non-zero, we always assume that the dispersive and diffusive fluxes are equal to zero.

The second type of boundary condition is the no-flux condition which is applied when there is no flow across the boundary. In this case, both the advective flux and the dispersive/diffusive fluxes are equal to zero.

The last type (often called the *first* type or a Dirichlet boundary condition) is normally applied when the advective flux is equal to zero but there is a dispersive or diffusive flux across the boundary (if the advective flux is non-zero, then the code automatically assumes a zero dispersive/diffusive flux). This condition is many cases is not physically reasonable, since it assumes that the diffusive and/or dispersive flux through the boundary does not affect the concentration of the exterior node (i.e., it is fixed). This boundary condition would normally be applied where the physical processes exterior to the domain are such that the diffusive/dispersive flux through the boundary is not enough to alter the concentrations there. One example might be a boundary along a fracture within which the flow is so rapid that the diffusive flux through the fracture wall has a negligible effect (e.g., Steefel and Lichtner, 1994). Another example might be a boundary corresponding to an air-water interface where a gas like O_2 or CO_2 is present in such large abundance that it effectively buffers the concentrations in the aqueous phase. Another example which is often mentioned in this regard is the case where mineral equilibria fixes the concentrations. This is dangerous, however, since the system is only completely determined in the special case where the phase equilibria results in an invariant point, i.e., no thermodynamic degrees of freedom exist. This is extremely rare in open systems, despite the frequent invocation of this condition in metamorphic petrology. Normally, since the concentrations will not be uniquely determined, a rigorous treatment would require that a speciation calculation with equilibrium constraints be carried out to actually determine what the concentrations at the exterior node should be. This, however, is not implemented at the present time.

OS3D and **GIMRT** differ slightly in their application of a Dirichlet boundary condition where the advective flux across the boundary is zero. In the case of **GIMRT**, the user may specify that an exterior node be a fixed concentration node and that it affect the interior node (i.e., there will be a non-zero diffusive and/or dispersive flux through the boundary). In **OS3D**, however, the code *always* assumes the dispersive/diffusive flux across the boundary is zero (due to the way the transport algorithm is formulated). If one wants to use a Dirichlet condition in the case of **OS3D**, one does so by fixing the concentrations at the first node or nodes interior to the boundary. This will have the same effect as allowing an exterior node influence the interior domain, except that one can think of the boundary as having been moved inward by one grid cell. The procedure for doing this is described in the section below explaining the input file `threedin`.

To summarize the boundary condition options in **OS3D** and **GIMRT**, a non-zero flow across the boundary means that the code will use the upstream, exterior boundary concentrations to give a purely advective flux at the boundary and the code will allow a fluid packet to pass out of the system with no influence from a downstream, exterior nodes. Exterior nodes across a boundary through which there is no flow will not influence the interior domain, unless the Dirichlet boundary condition option is used (usage described below). The default is therefore no diffusive/dispersive flux boundaries which is only overridden where the Dirichlet option is invoked.

Choice of Primary and Secondary Species

After setting up the problem physically, one needs to decide on the chemistry to be included. **GIMRT** requires that one specify an initial choice of primary species which then determine the number of independent chemical components in the system. This choice of primary species is not unique, however, as discussed in the section 2.2. The user must choose, however, which secondary species are to be included in the simulation. This can be an advantage for multicomponent reaction–transport modeling since one would like the option of carrying out chemically simplified simulations. However, the user must take care that important species are not neglected. Future versions of the code will have the capability of reading all of the potentially relevant species from the database.

The only cautionary note here is that one must include either among the primary or secondary species a species which is used in writing the reaction in the **EQ3/EQ6** data base. Note that this database is provided with the code in a reformatted form. For example, if one wants to include aluminum in the simulation, then the species Al^{+3} must appear among *either* the primary or secondary species since that species is used as the primary or basis species in the **EQ3/EQ6** data base. In the same way, one must include $O_{2(g)}$ among the list of gases if $O_{2(aq)}$ is to be present in the system since much of the **EQ3** database is written in terms of $O_{2(g)}$. If this is not done, the code will not be able to find the reaction when searching the data base. One should not, however, include any secondary species which cannot be written completely in terms of the basis species specified in the input file. The simplest way to grasp these features is to check the database to see which species are used to write the reactions involving a species of interest.

The species H_2O presents a special case since in many cases it is present in such abundance that its activity can be taken as unity. If one wants the activity of H_2O to be 1, then it should be left out of the list of primary and secondary species. The code will still balance the reactions properly as if H_2O were present. If, on the other hand, one wants to carry out a mass balance on H_2O, then it should be included in the list of primary species and it will be treated like any other species, except for the treatment of its activity coefficient. **GIMRT** assumes that the activity of H_2O is given by its mole fraction in the solution.

Redox Reactions

There are many possibilities for choices of primary and secondary species to represent redox reactions. **GIMRT** writes all redox reactions as whole cell reactions, i.e., electrons are *not* used as either primary or secondary species. It is easy to show that the electrons are completely unnecessary if the solutions are electrically neutral, and the inclusion of electrons requires that one include a “species" the mass of which should everywhere be zero. There are cases where electrons need to be included, as in corrosion problems where an actual electron flux occurs. When the solution is electrically neutral, however, no special provision for redox species and reactions is necessary when whole cell reactions are used (in contrast to statements by Yeh and Tripathi (1989)), although the inclusion of redox reactions implies that we have included an additional balance equation representing the conservation of exchangeable charge.

Since $O_{2(g)}$ is used as the basis species in redox reactions in the **EQ3/EQ6** data base, it must be included in the list of gases in the input file if redox reactions are to be considered. A gas, however, cannot currently be used as a primary species in **GIMRT** since gases are not included in the mass balances. One could use $O_{2(aq)}$, however, or any number of other species which form a redox couple, for example, Fe^{+2} and Fe^{+3}, or HS^- and SO_4^{-2}.

Acid- Base Reactions

Acid–base reactions require no special treatment other than in the initialization process where one normally fixes or calculates pH rather than an H^+ concentration. Once a pH has been either fixed or calculated in the initialization procedure, however, H^+ can be treated like any other component. Again, no special considerations are needed in contrast to the statements by Yeh and Tripathi (1989).

Activity Coefficient Model

At this time, the user may choose to run the code either with unit activity coefficients or with an extended Debye–Hückel formulation to calculate activity coefficients for the ionic species. The Debye–Hückel formulation is given by

$$\log \gamma_i = -\frac{AZ_i^2 (I)^{1/2}}{1 + B \overset{\circ}{a}_i (I)^{1/2}} + \dot{b}I \quad (63)$$

The coefficients for the temperature dependence of the parameters are taken from the **EQ3** data base.

Initialization Procedure

GIMRT initializes the solute concentrations at individual grid points over the domain by carrying out a distribution of species using a number of options. The initial concentrations of the primary species may be determined by

- carrying out a mass balance on a total concentration (e.g., aluminum),

- carrying out a charge balance on a species (e.g., Cl^-),
- a mineral equilibrium constraint (e.g., H^+ required to be in equilibrium with calcite),
- fixing the partial pressure of a gas (e.g., $CO_{2(g)}$),
- fixing the activity (e.g., pH) of a species,
- fixing the concentration of a species, and
- fixing the number of moles surface hydroxyl sites per mole mineral and the intrinsic surface area of the mineral (i.e., m^2/g).

This last option is available in **GIMRT** and it allows one to specify the concentration of surface complexes. These are specified by giving a site density (mole per m^2 mineral) and a surface area of mineral per gram mineral. The total concentration of surface hydroxyls developed on the mineral (converted to mol/kg solvent) is then computed from this information and the abundance of the mineral specified in the startup file. If a mineral is initially is not present but one would like to consider adsorption on this mineral, then the total concentration of surface hydroxyls on this mineral is set to a small number (10^{-10} mol/kg) and its concentration only builds up once the secondary mineral precipitates.

How each of the above initialization constraints is applied is described in the section on the input file below.

Temperature Gradients

Since **GIMRT** is based on finite difference methods, it has no problem handling temperatures which are non–constant either in time or in space. If either a non–default thermodynamic data base or the `master25.data` data base is not used at startup, then the code defaults to the `mastertemp.data` file where equilibrium constants are given at temperatures of 0°C, 25°C, 60°C, 100°C, 150°C, 200°C, 250°C, and 300°C at pressures corresponding to the water saturation curve. The code then fits a polynomial to the thermodynamic data so that equilibrium constants can be generated at any temperature between 0°C and 300°C along the water saturation curve. Simulations are possible at other pressures and temperatures, but the user must provide his or her own data base in the same format as `master25.data` or `mastertemp.data`. In addition, the algorithm for fluid density must be modified as well. Presently the fluid density is calculated as a function of temperature from a polynomial fit to the data in Helgeson and Kirkham (1974). Fluid viscosity is calculated from the data presented by Bruges and others (1966).

A temperature dependence is also included in the calculation of the reaction rate constants (Equation 30), the diffusion coefficients, and the fluid densities. Reaction rate constants at temperature are obtained normally by extrapolating the rate constants for 25°C given in the input file. If one would rather input directly the desired rate constant at temperature, than the simplest procedure (short of changing the source code) is to input the rate *as if it were the 25°C data* while changing the activation energy to 0 kcal/mole. There is also an option for directly inputting the diffusion coefficient at temperature.

There are two options for temperature presently implemented in **GIMRT**. The first is simply to run the simulation isothermally, either at 25°C or at temperature. The second is to specify a linear temperature gradient which operates only in the X direction. Many other ways of handling temperature are possible, including allowing for a transient temperature field, but this must be provided by the user/programmer at the present time. This can be done with a simple read of temperatures, making sure that the dimensions of the temperature field match those specified in the code. The positions in the code where a user should place a routine for calculating or inputting a temperature are marked in the routine `gimrt.f`.

Running CrunchFlow

CrunchFlow reads a user-provided input file on startup which provides the necessary physical and chemical parameters needed for a run. The input file, the name of which is specified by the user, is keyword-based so that the order of appearance does not matter. In many cases, it is possible to leave various optional parameters out altogether and thus allowing the code to use default parameters. Keywords are grouped broadly into keyword blocks, which in turn include a variety of keyword parameters. Keyword blocks may appear anywhere in the input file and keyword parameters may appear anywhere within a particular block, but certain keyword parameters must appear in the appropriate blocks. The possible keyword blocks include:

TITLE
RUNTIME
OUTPUT
PRIMARY_SPECIES
SECONDARY_SPECIES
GASES
MINERALS
AQUEOUS_KINETICS
ION_EXCHANGE
SURFACE_COMPLEXATION
PEST
DISCRETIZATION
INITIAL_CONDITIONS
BOUNDARY_CONDITIONS
TRANSPORT
FLOW
POROSITY
TEMPERATURE
CONDITION

With the exception of the keyword block **CONDITION**, the blocks should appear only once in the input file. Each keyword block is terminated by an **END**. The keyword block **CONDITION** is a special case in that it can occur multiple times. Each occurrence of **CONDITION** specifies a separate geochemical condition (these may be boundary or initial conditions or source terms) containing the geochemical input needed to describe a particular problem. The various keyword blocks will be discussed individually in more detail below.

Reading Input File Name from a File

Normally, CrunchFlow will prompt the user to enter the name of an input file interactively. The user can provide the name of the input file, however, by including it in a file names *PestControl.ant*. The code will check to see if this file exists in the directly from which CrunchFlow is invoked, and if it does, will attempt to read the file name from it. Upon successfully reading the input file name, the code will then check to see that this file exists before reading from it. The use of the *PestControl.ant* to provide an input file name will then skip the requirement of interactive input from the user, an option that is particularly useful when running PEST.

Keyword Blocks

Each keyword block is initiated with the appropriate keyword (case insensitive) on a line of its own and terminated by **END**, also on a line of its own. The contents of any keyword block, then, is anything between the keyword block name and the **END** specifier. Blank lines within the keyword block will be ignored. Lines beginning with **!** are treated as comments and ignored.

Space and Time Units

Keyword parameters involving space and time units are used in several of the keyword blocks. These take the form:

time_units	*unit*
space_units	*unit*

where *unit* in the case of time may be years (the default), days, hours, minutes, or seconds, and in the case of space may be meters (the default), kilometers, centimeters, millimeters, and micrometers. Non-metric units (e.g., feet) are not recognized. As with any other keyword parameter, they may occur anywhere within a particular block to which they apply. A specification of a space or time unit, however, is not global and applies only to the keyword block within which it occurs.

Space Units	Alternate Acceptable Specifications
meters (default)	meter, m
kilometers	kilometer, km
centimeters	centimeter, cm
millimeters	millimeter, mm
micrometers	micrometer, micron, microns, um

Table 1: List of possible spatial units which can be specified with the **space_units** keyword parameter.

Time Units	Alternate Acceptable Specifications
years (default)	year
days	day
hours	hour
minutes	minute
seconds	second

Table 2: List of possible time units which can be specified with the **time_units** keyword parameter.

Input File Formats

Various spatially-dependent field variables can be calculated outside of CrunchFlow and read in for use. These include the temperature, permeability, tortuosity, porosity, liquid saturation, and erosion/burial rates. Typically these are specified with a keyword (see below), followed by the file name. As of August 2006, it is also possible to specify the format of the file. The basic syntax is:

```
keyword     filename    InputFileFormat
```

where the keyword may be one of the following:

- read_VelocityFile
- read_GasVelocityFile
- read_PermeabilityFile
- read_PorosityFile
- read_SaturationFile
- read_TortuosityFile
- read_BurialFile
- read_TemperatureFile

The possible file formats include

- SingleColumn (the default)
- ContinuousRead
- FullFormat (dummy X and optionally Y and Z coordinates)
- Unformatted (binary)

SingleColumn Example

```
DO jz = 1,nz
  DO jy = 1,ny
    DO jx = 1,nx
      READ(52,*) tortuosity(jx,jy,jz)
    END DO
  END DO
END DO
```

ContinuousRead Example

```
READ(52,*) (qx(jx,jy,jz),jx=0,nx),jy=1,ny),jz=1,nz)
READ(53,*) (qy(jx,jy,jz),jx=1,nx),jy=0,ny),jz=1,nz)
READ(54,*) (qz(jx,jy,jz),jx=1,nx),jy=1,ny),jz=0,nz)
```

FullFormat Example

```
DO jz = 1,nz
  DO jy = 1,ny
    DO jx = 1,nx
      READ(52,*) xdum, ydum, zdum, tortuosity(jx,jy,jz)
    END DO
  END DO
END DO
```

Unformatted Example

READ(52) qx

TITLE

The TITLE keyword block is used to provide a title for a particular simulation. It is echoed to the standard output during a run and written to the output file crunch.out. The title will be read as a character string up to 264 characters long.

Example:

```
TITLE
Lichtner's 2D copper leaching problem
END
```

RUNTIME

The **RUNTIME** keyword block contains a number of parameters for running a particular simulation. All of them are optional, as is the **RUNTIME** keyword block itself, since defaults values will be used if none are provided by the user. All of the parameters, with the exception of the species name given in the keyword parameter **master**, are case insensitive. Within the **RUNTIME** keyword block, the following keyword parameters are possible.

coordinate
correction_max
courant_number
database
database_sweep
debye-huckel
density_module
dissolution_max
fix_saturation
generic_rates
gimrt
gimrt_pc
gimrt_pclevel
gimrt_solver
graphics
hindmarsh
lag_activity
later_inputfiles
master
overshooto\tolerance
pc
pc_level
precipitation_max
reaction_path
read_saturationfile
restart
save_restart
screen_output
solver
speciate_only
steady-state
timestep_max
timestep_init
time_tolerance

These keyword parameters are described in detail below.

Example:

```
RUNTIME
 time_units           years
```

```
timestep_init       1.E-09
timestep_max        0.01
time_tolerance      0.05
gimrt               true
debye-huckel        true
database_sweep      false
speciate_only       false
master              H+
screen_output       10
END
```

Coordinate

Keyword followed by an option, either *rectangular, cylindrical,* or *spherical*

Syntax: coordinate option

Default: Rectangular

Explanation: Rectangular coordinates are the default, but the user may also choose cylindrical or spherical coordinates.

Correction_max

Absolute magnitude of the maximum correction to the natural logarithm of a component species in any one Newton step.

Syntax*:* **correction_max** *value*

value is a positive real number

Default: 2.0

Explanation: This is an optional parameter which controls the absolute magnitude of the maximum allowable correction to a component species in any particular Newton step. The correction is in units of the natural logarithm of the species molality. The limitation of the correction is a form of damping of the Newton step which prevents overflows in many cases. Correction factors over 10 are not recommended. Excessively small correction factors (< 1) will slow the rate of convergence of the Newton method significantly.

Courant_number

Maximum Courant or Courant-Friedrichs-Lewy number (defined by $CFL = {}^{v\Delta t}\!/\!_{\Delta x}$) for OS3D option (GIMRT *false)*

Syntax*:* **Courant_number** *value*

value is a positive real number between 0.0 and 1.0

Default: 0.5

Explanation: This is an optional parameter that controls the maximum CFL number by limiting the timestep, Δt, when the OS3D option is used. The keyword is ignored if the GIMRT option is selected. The Courant number cannot be greater than 1.0 for the OS3D option, since results in an unstable method.

Database

Option to specify a database file.

Syntax: **database** *[filename]*

filename is an optional string (up to 132 characters). indicating the name of the database file.

Default: `datacom.dbs`

Explanation: This specification of the name of the database file may also go in the DATABASE keyword block.

Database_sweep

Keyword followed by *true* or *false* which selects whether to carry out a full sweep of the database, loading all possible species and minerals.

Syntax: database_sweep logical

logical is a standard FORTRAN logical (true or false).

Default: false

Explanation: The keyword parameter database_sweep is followed by true or false (or yes or no) and is used to select, when true, the option to sweep the entire thermodynamic database so as to load all possible species and minerals. In this case, the user's choice of secondary species and minerals listed in the input file is ignored. After the sweep, each of the geochemical conditions is speciated and the code stops. User's who want to use these species and minerals in an actual reactive transport run need to copy the list of secondary species and/or minerals over to the keyword blocks for these categories within the input file and then deselect the database_sweep option. In this mode, the code runs essentially like EQ3 does in terms of loading all possible relevant species given a choice of primary component species. This option supersedes whatever is set with the speciate_only keyword described below.

Debye-Huckel

Keyword followed by *true* or *false* which selects model for calculation of activity coefficients.

Syntax: debye-huckel logical

logical is a standard FORTRAN logical (true or false).

Default: true

Explanation: The keyword parameter debye-huckel is followed by true or false (or yes or no) and is used to select the model to compute activity coefficients. Currently, the only two options are the extended Debye-Huckel model (Helgeson) or activity coefficients equal to 1 (no activity corrections).

Density_module

Keyword followed by an option, either temperature, sodium_nitrate, sodium_chloride, or potassium_nitrate.

Syntax: density_module option

Default: Temperature

Explanation: The default is to calculate the fluid density only as a function of temperature. Limited options are available to calculate the fluid density as a function of composition, with sodium nitrate and sodium chloride solution being the two most common.

Dissolution_max

Maximum decrease in mineral volume fraction for any one time step.

Syntax: dissolution_max value

value is a real positive number.

Default: 0.001 (m^3 mineral/m^3 porous medium)

Explanation: This keyword parameter specifies the maximum decrease in mineral volume fraction (dimensionless) for any one time step. This option is applied in calculating the size of the time step for the next time step based on rates computed in the preceding one. It is not, therefore, used currently as a firm control on time step since no repeat of a time step is carried out if dissolution_max is exceeded in a step.

Fix_saturation

A constant, uniform liquid saturation may be specified with this keyword.

Syntax: fix_saturation *value*

value is a positive real number between 0.0 and 1.0

Default: 1.0

Explanation: This is an optional parameter that can be used to set a constant, uniform liquid saturation for a problem. The value must be greater than 0.0 and less than or equal to 1.0.

Generic_rates

Keyword followed by a value representing the logarithm of the reaction rate to be used for all minerals in the MINERAL keyword block. This option avoids the need for kinetic entries for the minerals in the database

Syntax*:* **generic_rates** *value*

value represents the logarithm of the mineral rate in units of mol/m^2/sec.

Default: The option is not selected and individual kinetic entries will be required in the database.

Explanation: The keyword parameter **generic_rates** is followed by a value for the logarithm of the mineral reaction rate that will be used for all of the minerals given in the MINERAL keyword block. In this mode, the code will not require that a kinetic rate law be explicitly given in the database for each of the minerals considered. This option may be useful where the interest is in simply tracking a reaction path without detailed kinetic information.

Gimrt

Keyword followed by *true* or *false* which selects coupling method for reaction and transport.

Syntax*:* **gimrt** *logical*

logical is a standard FORTRAN logical (true or false).

Default: true

Explanation: The keyword parameter **gimrt** is followed by *true* or *false* (or *yes* or *no*) and is used to select a global implicit coupling of reaction and transport (GIMRT) if true, or time splitting of reaction and transport (OS3D) if false. Other restrictions on the coupling method may apply—for example, a global implicit method of coupling reaction and transport is currently allowed only up to 2 dimensions. The selection of a global implicit method (**gimrt** *true*) eliminates any strict requirement that a CFL criteria be observed, although other practical limitations on the time step (either in terms of numerical stability or of time truncation error) may apply. See Steefel and MacQuarrie (1996) for a fuller discussion of the pros and cons of the two coupling methods.

Gimrt_pc

Keyword followed by an option, either *bjacobi,* or *ilu.*

Syntax*:* **gimrt_pc** *option*

Default: *bjacobi*

Explanation: This sets the preconditioner method for the GIMRT option, with the choices being a block Jacobi method (bjacobi) or ILU. See the PETSc web pages at www.anl.gov/petsc for further explanation.

Gimrt_pclevel

Keyword followed by an integer value giving the level of fill to be used in preconditioning the global implicit reactive transport matrix.

Syntax*:* **gimrt_pclevel** *integer value*

Default: *2*

Explanation: This sets the preconditioner level of fill for the GIMRT option as an integer value. Using higher levels of fill than 2 may slow the speed of execution. See the PETSc web pages at www.anl.gov/petsc for further explanation.

Gimrt_solver

Keyword followed by an option, either *gmres* or *bcgs.*

Syntax*:* **gimrt_solver** *option*

Default: *gmres*

Explanation: This sets the solver method for the GIMRT option, with the choices being the GMRES method (*gmres*) or a stablized biconjugate gradient (*bcgs*). See the PETSc web pages at www.anl.gov/petsc for further explanation.

Graphics

Keyword followed by an option, either *kaleidagraph*, *tecplot,* or *xmgr*.

Syntax*:* **graphics** *option*

Default: *kaleidagraph* for *1D, tecplot* for 2-3D.

Explanation: This sets the style of output for the various graphics packages, with *Kaleidagraph* being the default for 1D, *Tecplot* the default for 2D. The *xmgr* option can be used for either the X windows based *xmgr*, or for *gnuplot*.

Hindmarsh

Keyword followed by *true* or *false* which selects the linear solver for one-dimensional problems.

Syntax*:* **hindmarsh** *logical*

logical is a standard FORTRAN logical (true or false).

Default: true

Explanation: The keyword parameter **hindmarsh** is followed by *true* or *false* (or *yes* or *no*) and is used to select the linear solver for one-dimensional problems. Setting **hindmarsh** as true turns on the direct block tridiagonal solver written by Alan Hindmarsh (1977). Setting **hindmarsh** to false turns on the PETSc solver.

Lag_activity

Keyword followed by *true* or *false* which selects when activity coefficients are updated.

Syntax*:* **lag_activity** *logical*

logical is a standard FORTRAN logical (true or false).

Default: true

Explanation: The keyword parameter **lag_activity,** when true, causes the code to update activity coefficients only at the beginning of the time step, outside of the Newton iteration loop. In this scheme, activity coefficients are based on the ionic strength calculated from the previous time step. Since at this time ionic strength is not included as an independent unknown in CrunchFlow, the method yields accurate results in most cases and shows better convergence behavior than does a scheme in which activity coefficients are updated throughout a particular Newton iteration loop. The method may not be completely accurate when large contrasts in ionic strength occur in a problem, although even this depends on the time step used.

Later_inputfiles

Option to specify additional input files to be read on restart.

Syntax*:* **later_inputfiles** *[filename1],[filename2],[filename3]…*

filename1, filename2… are strings (up to 132 characters). indicating the name of the input files to be read following restart.

Default: No additional input files are read.

Explanation: This option allows additional input files to be read automatically restart. Normally this option is used in conjunction with the keywords *save_restart* and *restart*, which generate and read respectively restart of “pickup” files that can be used to read in the spatial distribution of species and minerals from a previous run, changing a boundary condition. There is no a a priori restriction on the number of files. The *later_inputfiles* option should be specified only in the first input file in the series to be read (later specifications will be ignored).

Master

Master variable to be used for time step control.

Syntax*:* **master** *name*

name is a string up to 132 characters representing the name of a species.

Default: *O2(aq),* if present, otherwise H^+ if present, otherwise species number 1.

Explanation: The keyword parameter time_tolerance is applied to this species. Normally, for redox problems *O2(aq)* is used since it is the most sensitive “master variable”. For non-redox multicomponent problems, H^+ is normally used. The name of the species is case-sensitive, unlike other parameters specified within the **RUNTIME** keyword block.

OvershootTolerance

Keyword followed by a real number giving the tolerance for overshooting a mineral volume fraction.

Syntax*:* **OvershootTolerance** *value*

value is a positive real number

Default: *1.0E-05*

Explanation: This keyword sets the tolerance for overshooting a zero volume fraction. This is necessary because of the way in which CrunchFlow handles mineral abundances. For any one time step, mineral abundances and surface areas are assumed fixed—mineral volume fractions, therefore, are updated with an explicit scheme at the end of the time step. This approach, however, makes it possible to overshoot the available mineral concentration, which can result in overall mass balance errors. The tolerance set here specifies the maximum overshoot in units of m^3 mineral/m^3 porous medium. If the tolerance is exceeded, the time increment, *delt*, is cut and the time step is repeated.

Pc

Keyword followed by an option, either *ilu*, *jacobi*, or *direct.*

Syntax*:* **pc** *option*

Default: *ilu*

Explanation: This sets the preconditioner method for PETSc to solve the flow or diffusion equations. It does not apply to the solution of the global implicit reactive transport equations in the GIMRT option. The choices include *ilu*, with a partial fill set by the keyword *pclevel*, *jacobi*, and *direct*. The *direct* option carries out a direct solution of the matrix without iteration (i.e., it is not a sparse matrix solver). See the PETSc web pages at www.anl.gov/petsc for further explanation.

Pclevel

Keyword followed by an integer value giving the level of fill to be used in preconditioning the flow or diffusion matrices.

Syntax: **pclevel** *integer value*

Default: *5*

Explanation: This sets the preconditioner level of fill for solving the flow or diffusion equation. An integer value must be provided. Normally, levels of fill of above 5 should only be used in the case of an ill-conditioned matrix, as typically occurs with large differences in permeability. Use of higher fill levels will slow execution in the case of reasonably well-conditioned linear systems. See the PETSc web pages at www.anl.gov/petsc for further explanation.

Reaction_path

Keyword followed by *true* or *false* which selects whether or not to run in "reaction path" mode as opposed to "batch" mode.

Syntax: **reaction_path** *logical*

logical is a standard FORTRAN logical (true or false).

Default: false

Explanation: The keyword parameter **reaction_path** is followed by *true* or *false* (or *yes* or *no*) and is used to select, when true, the option to run in "reaction path" mode. This mode is contrasted with "batch" mode and applies only in 0 zero spatial dimension problems (a single grid cell is used). In reaction path mode, mineral volume fractions are not updated. This is meant to represent (loosely) the case of pure advective transport in which secondary mineral products are left behind in a flow path.

Read_SaturationFile

Option to specify a file with liquid saturations to be read.

Syntax: **read_SaturationFile** *filename format*

filename gives the name of the file(up to 132 characters) contining the liquid saturation distributed over the entire spatial domain.

Default: None.

Explanation: This provides the name of a file containing liquid saturations distributed in space to be read. This option overrides the *fix_saturation* keyword. The format of the file depends on the optional format specified (see Section 7.1.2). If no format is specified, the code assumes a single column (SingleColumn) of values in the order 1-NX, 1-NY, 1-NZ:

```
DO jz = 1,nz
  DO jy = 1,ny
    DO jx = 1,nx
```

```
          READ(52,*) satliq(jx,jy,jz)
        END DO
      END DO
    END DO
```

Restart

Option to use a restart file.

Syntax*:* **restart** *[filename] [append]*

filename is an optional string (up to 132 characters). indicating the name of the restart file.

Default: crunch.rst

Explanation: This keyword parameter specifies that the code should be restarted. If restart is not followed by anything, the default restart file name of *crunch.rst* is used. Alternatively, the user may specify an optional restart file name (this can be useful, since the default file name will be overwritten with a new run). This optional restart file, however, must have been created in an earlier run with the *save_restart* keyword. Binary restart files will be created each time a set of spatial profiles are written (see keyword *spatial_profile* in the **OUTPUT** keyword block). With a normal successful termination of the code, the restart file will be updated at the end of the run. Subsequent output to the "breakthrough" files (see keyword *time_series*) can be appended by adding the optional *append* keyword after the filename. In the same way, the file counter "n" (for example, pHn.dat) will be updated as well if this *append* option is chose, so spatial profiles written after restart will reflect the files written previously before restart.

Save_restart

Option to save a restart file.

Syntax*: save*_**restart** *[filename]*

filename is an optional string (up to 132 characters). indicating the name of the restart file to be written to.

Default: *crunch.rst*

Explanation: This keyword parameter specifies that the code should create a restart file with the name provided. If the keyword *save_restart* is not provided, restart information will be saved to the default restart file name of *crunch.rst*.

Screen_output

Keyword followed by an integer value giving the interval at which run time output is written to the screen.

Syntax*:* **screen_output** *integer value*

Default: *1*

Explanation: This gives the interval at which run time output (time, time step size, number of Newton iterations required…) is written to the screen. This is useful where the code is taking a large number of time steps to complete a run and only periodic information is needed. This number should be a minimum of 1 (output every time step).

Solver

Keyword followed by an option, either *bicg*, *gmres*, *bcgs*, or *direct.*

Syntax*:* **solver** *option*

Default: *bicg*

Explanation: This sets the solver method for PETSc to solve the flow or diffusion equations. It does not apply to the solution of the global implicit reactive transport equations in the GIMRT option. The choices include a bi-conjugate gradient solver, *bicg*, and the additional options of *gmres, bcgs*, and *direct*. The *direct* option carries out a direct solution of the matrix without iteration (i.e., it is not a sparse matrix solver). See the PETSc web pages at www.anl.gov/petsc for further explanation.

Speciate_only

Keyword followed by *true* or *false* which selects whether or not to speciate the geochemical conditions and then stop.

Syntax*:* **speciate_only** *logical*

logical is a standard Fortran logical (true or false).

Default: false

Explanation: The keyword parameter **speciate_only** is followed by *true* or *false* (or *yes* or *no*) and is used to select, when true, the option to speciate the geochemical conditions and then to stop without carrying out any reactive transport calculations. This option is contrasted with database_sweep which, in addition to stopping after the speciation calculations, also sweeps the database to load all of the possible species and minerals. Using speciate_only, the user's list of secondary species and minerals given in the input file is used.

Steady_state

Keyword followed by *true* or *false* which selects whether or not to run to a quasi-steady state as a condition for stopping a particular simulation.

Syntax*:* **steady_state** *logical* [*tolerance*]

logical is a standard FORTRAN logical (true or false)

tolerance is an optional real number giving the criteria for attainment of a steady state.

Default: false, with a tolerance of 1.E-07 in units of mol/kg/year normalized to the primary species total concentration (so tolerance has units of 1/year).

Explanation: The keyword parameter **steady_state** is followed by *true* or *false* (or *yes* or *no*) and is used to select, when true, the option to run to a quasi-steady state and then terminate the program rather than to rely on the specific output times given in the **OUTPUT** keyword block. If true, the logical specifier can be followed by an optional convergence criteria in units of normalized mol/kg/year for all of the primary species total concentrations (the normalization then results in a convergence criteria with units of 1/year). Depending on the value of the convergence criteria (a value of 1.E-07 is the default), a quasi-steady state may be attained since in many cases slow dissolution of primary mineral phases may prevent a true steady-state from being achieved. In many cases, however, it is possible to attain a "quasi-stationary state" (see Lichtner, 1988) in which aqueous concentrations are nearly unchanging while mineral volume fractions are slowly evolving.

Timestep_max

Indicates maximum time step allowed in problem.

Syntax*:* **timestep_max** *value*

value is a positive real number

Default: 1.0 year

Explanation: The default units are years, with the option to reset the time units to one of those discussed under **Time Units** (e.g., seconds, days, etc.) using the **time_units** parameter keyword. This is the maximum allowed time step, but a smaller maximum may be used if dictated by Courant number restrictions, as in the case of CrunchFlow run in OS3D mode (time splitting of transport and reaction).

Timestep_init

Indicates initial time step to be used in simulation..

Syntax*:* **timestep_init** *value*

value is a positive real number

Default: 1.E-09 years

Explanation: The default units are years, with the option to reset the time units to one of those discussed under **Time Units** (e.g., seconds, days, etc.) using the **time_units** parameter keyword. The initial time step also serves as the minimum time step which is only superseded by a restriction on the Courant-Friedrich-Lewy (CFL) number.

Time_tolerance

Tolerance for the second derivative of the change in the natural logarithm of the master species with respect to time.

Syntax*:* **time_tolerance** *value*

value is a positive real number

Default: 0.001

Explanation: Tolerance applies to the second derivative of the natural logarithm of the master species (specified by the keyword parameter master). This parameter can be used to either control the time truncation error or to limit the increase in time steps for the sake of numerical stability. With a time tolerance set very high, the code will normally increase the time step by a factor of 2 every time step until the maximum time step is reached.

OUTPUT

The keyword block **OUTPUT** controls when and what will be written to disk. If the OUTPUT block is not given, then no time stepping will occur (see keyword parameter **spatial_profile**). The keyword parameters included in the OUTPUT keyword block are:

time_units
spatial_profile
time_series
time_series_print
time_series_interval
MakeMovie

The *time_units* keyword parameter has the same definition and role as in other keyword blocks where it appears. It specifies the time units for such keyword parameters as *spatial_profile* and it determines the units used in the various output files.

Example:

```
OUTPUT
time_units              hours
spatial_profile     250
time_series         50
time_series         100
time_series_print       Cs+  pH
time_series_interval 1
END
```

In this example, a single spatial profile (field variables as a function of space) is written at 250 hours. The code will time step until 250 hours is reached and then stop. A total concentration of cesium will be tracked at JX=50 and JX=100 (nodes 50 and 100 in the X direction). Output to the time series file will be every time step. Since this is a 1D problem, the Y and Z coordinates need not be given.

Spatial_profile

Option to write a spatial profile or “snapshot” at the specified time(s).

Syntax*:* **spatial_profile** *time***1** *[time***2** *time***3** *… &*

*time***n***]*

time1, time2, etc. are real numbers giving the time at which the spatial profile is to be written. The ampersand can be used to continue listing of output times on multiple lines. Any one line can be up to 132 characters long.

Default: If not specified, the code assumes that no output file is to be written and therefore no time stepping will occur.

Explanation: This keyword parameter instructs the code to output a spatial profile or “snapshot” at the specified time. The time units are taken from the *time_units* value specified in the **OUTPUT** keyword block (if absent, time units of years are assumed). An ampersand can be used to continue listing of output times on the following line. This important parameter, in addition to specifying when spatial profiles will be written, controls whether time stepping is carried out at all. The code will run until the last output time in the list is reached, unless the parameter *steady_state* in the **RUNTIME** keyword block is turned on (in this case, the code runs until a steady-state is achieved **or** until the final spatial profile time is reached, whichever is first). Field variables written out as part of the spatial profile are given in the **Output Variables** section below.

Time_series

Option to specify nodal location of a time series.

Syntax*:* **time_series** filename1 *JX [JY JZ]* for a 1D problem

time_series filename1 *JX JY [JZ]* for a 2D problem

time_series filename3 *JX JY JZ* for a 3D problem

where *filename#* is the file name to be used for the time series, and *JX, JY,* and *JZ* are integers giving the node number for which the time series will be tracked.

Default: There are no default values.

Explanation: This keyword parameter specifies that a time series should be recorded at the node specified. Multiple specifications of *time_series* may occur, each one giving a different filename and node number. For one-dimensional problems, the Y and Z coordinates are optional. If in any case a node location is given that is greater than the coordinates specified in the **DISCRETIZATION** block, execution will terminate and an error statement will be written (e.g., *JX* > *NX*, the number of nodes in the X direction).

Time_series_print

Option to write out a time series for one or more solutes..

Syntax*:* **time_series_print** *species**1** [species**2** species**3** … &]*

*[… species**n**]*

where *species**n*** is a character string corresponding to an aqueous species name.

Default: If *time_series_print* is not specified, no tracking of solutes will be done. If *time_series_print* is not followed by a species name, a default of **all** will be assumed.

Explanation: This keyword parameter gives the species to be tracked as a time series. The *time_series_print* keyword may be followed with *all*, in which case all of the total concentrations will be written out in column form in units of mol/kg followed by all of the species concentrations in logarithmic form. Listing of species names will cause the total concentrations of these species to be written out (not the individual species concentrations). In addition, pH may be specified to be output

As an example, the following line

```
 time_series_print  pH  Ca++  Na+  K+
```

will cause the code to track the time evolution of the solution pH and the total concentrations of Ca, Na, and K. Species names must match those in the primary species list (case sensitive).

Time_series_interval

Time step interval at which a time series should be output.

Syntax*:* time_series_interval *interval*

where *interval* is an integer.

Default: 1

Explanation: This keyword parameter gives the time step interval at which the time evolution of the species specified in *time_series_print* will be written out at the nodal location specified by *time_series* keyword.

Time_series_output

Specific times at which a time series should be output.

Syntax*:* time_series_output *time1 time2 time3 …*

where *time1 and time2* are real numbers.

Default: None

Explanation: This keyword parameter gives the option of writing out time series at specific times. This is useful when the objective is to compare discrete data (e.g., from a kinetic experiment) with output from the simulation.

MakeMovie

Option to write out a movie file of specified total concentrations.

Syntax*:* *`MakeMovie species1 species2` …*

where *species1* and *species2* are primary species.

Default: No movie file will be written

Explanation: This keyword parameter provides the option to write out a total concentration for a primary species in 1D or 2D. The interval at which data is written out is given by *time_series_interval*.

DISCRETIZATION

This keyword block is used to set up the structured grid for a particular problem. The code assumes (presently) that rectilinear coordinates are used. The three keyword parameters which are recognized are:

```
xzones
yzones
zzones
```

each being used to set the grid dimensions in one coordinate direction.

Xzones

Discretization in the first (X) direction.

Syntax*:* **xzones** *#cells spacing #cells spacing … & #cells spacing*

where *#cells* is an integer and *spacing* is a real number.

Default: *1 1.0* if **xzones** is not specified.

Explanation: This keyword parameter specifies the discretization in the X (first) coordinate direction. In one-dimensional problems, the X coordinate rather than the Y or Z coordinate should be used. The discretization is given by specifying pairs of integers and real numbers which give the number of cells of a given spacing. The following example would result in 10 cells of 1 meter spacing, 10 cells of 2 meter spacing and 10 cells of 4 meter spacing:

xzones *10 1.0 10 2.0 10 4.0*

resulting in a total of 30 cells in the X direction. Lines may be continued on the following line by using an ampersand (“&”) at the end of the line as a continuation marker.

Yzones

Discretization in the first (Y) direction.

Syntax*:* **yzones** *#cells spacing #cells spacing … & #cells spacing*

where *#cells* is an integer and *spacing* is a real number.

Default: *1 1.0* if **yzones** is not specified.

Zzones

Discretization in the first (Z) direction.

Syntax*:* **zzones** *#cells spacing #cells spacing … & #cells spacing*

where *#cells* is an integer and *spacing* is a real number.

Default: *1 1.0* if **zzones** is not specified.

PRIMARY_SPECIES

The **PRIMARY_SPECIES** keyword block is used to specify the primary component species for a particular problem. Borrowing from the tradition of EQ3/EQ6, there is a list of “database primary species” given in the first section of any database file. The list of “database primary species” serves as the building blocks of reactions in the main part of the database. From the point of view of CrunchFlow, however, primary species are any that are so designated in the input file. The user may list any mathematically valid set of primary species which spans the concentration space for a particular problem. If aqueous kinetic reactions are included in the problem, then these must be made up entirely of the user’s choice of primary species (not necessarily those found in the database). It should be noted, however, that some choices of primary species will result in non-intuitive components, making it difficult to specify initial and boundary conditions for a particular problem.

The primary component species (or “basis species”) are the chemical building blocks for a particular problem. The concentrations of the primary species are the major unknowns in the chemistry part of a reactive transport problem. The partial differential equations for the conservation of chemical mass in the system are written in terms of the total concentration of these primary component species. This formulation, which has been widely discussed by a number of workers, is used instead of equations for the conservation of the various chemical elements.

Database Format

"Database primary species" are listed in the database immediately following the specification of the temperature field (first line) and 3 lines giving the temperature-dependent Debye-Huckel parameters. The primary species block in the database is terminated with the line:

'End of primary' 0.0 0.0 0.0

The "database primary species" are specified in the following form:

<'SpeciesName'> <Debye-Huckel size parameter> <Charge> <Molecular weight>

Each "database primary species" is given on a separate line in the database in free format. The species name is enclosed in single quotes.

Input File Entry of Primary Species

The format of the **PRIMARY_SPECIES** keyword block is simple, consisting of a list of species name to be used. In contrast to other keyword parameters in the input file, the species names are case-sensitive. This is to remove the ambiguity which may arise from various species and compounds have similar names (for example, carbon monoxide or CO and cobalt or Co).

The list of species in the **PRIMARY_SPECIES** keyword block is important in that it defines the chemical space for a particular problem (thus the term "component species"). When geochemical conditions are given, all of the primary species listed in the **PRIMARY_SPECIES** keyword block must be specified in each of the conditions. In this respect, **CrunchFlow** differs from a code like **PHREEQC** (Parkhurst, 1995) where solutions may be mixed which have different sets of non-zero chemical concentrations specified (**PHREEQC** presumably sets unmentioned concentrations to 0). Since **CrunchFlow** grew out of a standard finite-difference approach to solving partial differential equations rather than out of reaction path approaches, it requires a consistent set of primary species for each cell within the domain and for the system boundaries.

The two examples below represent two different basis sets of primary species which can be used to solve the same problem. They are mathematically equivalent, therefore, given the correct choice of constraints. The total concentrations of some of the species will not change from one representation to the other—for example, total HCO_3^- will equal total CO_2(aq). However, other total concentrations will change, including that of H^+. Use of the Fe^{++}/Fe^{+++} redox couple is an alternative way of handling redox than is given by the use of Fe^{++} and O^2(aq). Note that the definition of total elemental Fe will change depending upon which formulation is used.

Example 1:

```
PRIMARY_SPECIES
Na+
K+
Ca++
H+
Al+++
Fe++
SiO2(aq)
HCO3-
SO4--
Cl-
O2(aq)
END
```

Example 2:

```
PRIMARY_SPECIES                    SiO2(aq)
Na+                                CO2(aq)
K+                                 SO4--
Ca++                               Cl-
H+                                 Fe+++
Al+++                              END
Fe++
```

SECONDARY_SPECIES

The **SECONDARY_SPECIES** keyword block gives the user's choice of secondary species for a particular problem. Secondary or "non-component" species are those for which an equilibrium reaction relationship is assumed with the primary species in the problem. In other words, the secondary species can be written in terms of the set of primary species. While the secondary species could be viewed as any species for which there is a reaction relationship with the primary species, whether the reaction is at equilibrium or kinetically controlled, in this code the secondary species are used to imply on species where the reaction is assumed to be at equilibrium. When a particular reaction is at equilibrium, a secondary species can be written algebraically in terms of the primary species concentrations and the equilibrium constant for the reaction using a mass action expression. Thus, the species can be mathematically eliminated as an independent unknown from the conservation equations. In contrast, where the reaction is kinetically controlled, a differential (as opposed to algebraic) equation is needed to relate the species to the other species in the problem.

Database Format

As with the primary species described above, a distinction must be made between secondary species in the database versus secondary species specified by the user in the input file. The "database secondary species" block follows the primary species block and is initiated with the line marking the end of the "database primary species" block:

```
'End of primary'   0.0   0.0   0.0
```

and terminated by

```
'End of secondary' 1 0. '0' 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
```

"Database secondary species" are written in every case as the destruction one mole of the secondary species. Using the standard convention, therefore, the stoichiometric coefficient for the secondary species is assumed to be –1, even though this is not explicitly given in the database. The format for a "database secondary species" is:

<'SpeciesName' > <Number of species in reaction (integer) > <Stoichiometric coefficient> <'SpeciesName'> <Stoichiometric coefficient> <'SpeciesName'> …<Log K array> <Debye-Huckel size parameter> <Charge> <Molecular weight>

In this format, the number of pairs of stoichiometric coefficients and species names is determined by the preceding value in "Number of species in reaction" (if there is a mismatch, a read error will result). The length of the "Log K" array is given by the number following 'temperature points' in the first line of the database file. Normally, eight temperature points are used following the EQ3 database format, for example:

'temperature points' 8   0. 25. 60. 100. 150. 200. 250. 300.

In this example, log K values for the reaction will be provided at eight temperatures between 0C and 300C.

An example of an entry for the species CO2(aq) is given by:

```
'CO2(aq)'  3   -1.00 'H2O'  1.00 'H+'  1.00 'HCO3-' -6.5804  -6.3447 -
6.2684  -6.3882  -6.7235  -7.1969  -7.7868  -8.5280  3.0  0.0   44.0098
```

Input File Entry of Secondary Species

The secondary species list provides the additional aqueous species used in a reactive transport problem. If the user selects the *database_sweep* option in the **RUNTIME** keyword block, however, the code will automatically load all possible secondary species, ignoring the user's list. However, when this option is selected, the code will stop after speciating the geochemical conditions. To use the set of secondary species generated by the *database_sweep* option, the user needs to formally copy and paste the secondary species list into the **SECONDARY_SPECIES** keyword block and turn off the *database_sweep* option. The list of secondary species generated by the database sweep are found in the output file "'input filename prefix'.out" which is generated every time a run with CrunchFlow is carried out. This is done because of the large number of potentially extraneous species which may be loaded for a particular problem, especially in the case of organics. The disadvantage of the approach is that the user may potentially neglect some important secondary species (i.e., those which have non-trivial concentrations in a particular problem). The recommended procedure is to begin a problem by running the code in *database_sweep* mode and then winnowing the list of secondary species *if necessary*. In most one-dimensional problems, the full list of secondary species may be used (with the exception of problems where a huge list of organic species may be loaded).

The format for listing secondary species in the input file is the same as the format for primary species. Secondary species to be included are listed on separate lines within the **SECONDARY_SPECIES** keyword block. If the database sweep option has been chosen previously, secondary species written to the file "'input filename prefix'.out" may be copied and then pasted into the input file. As with primary species, the input is case-sensitive—the species listed must be found in the database or an error message will result.

Example:

```
SECONDARY_SPECIES
OH-
AlOH++
HCO3-
END
```

GASES

The **GASES** keyword block specifies the user's choice of gases for a particular problem. In the case of fully saturated flow, gases are only used in the initialization procedure at the present time since their mass is not explicitly accounted for. Future releases of the code will have an option to specify a kinetic reaction with a gas phase which could be used, for example, to continually equilibrate a solution with the atmosphere. Clever users will recognize that the same effect can be obtained presently by specifying a fictional "mineral" in the **MINERAL** keyword block whose equilibrium constant corresponds to the aqueous concentration at which a species would be in equilibrium with a particular partial pressure of the gas phase (an example is given below).

During the initialization of the various geochemical conditions, the option is available for equilibrating the solution with one or more gases at specified partial pressures. In previous versions of the code, it was necessary to input the gas O2(g) in order to get most of the redox reactions to load properly. In the current version, however, the specification of O2(g) is optional since the code will automatically load it if the aqueous species O2(aq) is found in the primary or secondary species list. For fully saturated problems, therefore, gases need to be input only if they are used in constraining a geochemical condition.

For unsaturated transport and gas-liquid partitioning, gases must be explicitly entered. In the case of unsaturated transport and reaction, an actual mass balance on the individual gas concentrations is carried out.

If the *database_sweep* option is specified, then the code will automatically sweep the thermodynamic database looking for relevant gases based on the user's choice of primary species. Any entries in the **GASES** keyword block will be ignored in this case. These will then be listed in the standard output file "'input filename prefix'.out".

Database Format

The format for gases in the database is similar to that of the secondary species:

<'Gas' > <Molar volume (not currently used by code)> <Number of species in reaction (integer) > <Stoichiometric coefficient> <'SpeciesName'> <Stoichiometric coefficient> <'SpeciesName'> …<Log K array> <Molecular weight>

Input File Entry of Secondary Species

Gases are input as a simple list, with one gas per line of the **GASES** keyword block. If the database sweep option has been chosen previously, the list of gases written to the file "'input filename prefix'.out" may be copied from this file and then pasted directly into the **GASES** block.

MINERALS

The keyword block **MINERALS** refers to any solid-aqueous phase heterogeneous reaction, mineral-water reactions being the most common of these. The keyword parameters for this block are simply the names of the solid phases involved in the reaction. CrunchFlow currently assumes in all cases that the reaction involves the dissolution of one mole of the mineral or solid phase in question (that is, it's stoichiometric coefficient is –1). Solid phase reactions specified in an input file involve two separate entries in the database file: 1) a *thermodynamic* entry which gives the stoichiometry of the reaction, the equilibrium constants as a function of temperature, the molar volume of the solid phase, and its molecular weight, and 2) a *kinetic* entry which gives the rate law and rate coefficients for the reaction. All mineral or solid-aqueous phase reactions are considered to be kinetic in CrunchFlow, except in the initialization procedure where equilibrium with a solid phase may be selected. In practice, equilibrium with respect to solid phases is attained by choosing rate coefficients that are large relative to transport rates.

Database Formats

Thermodynamic Database Entries

The thermodynamic database for solid-liquid phase reactions begins with a separate line for each entry, with the mineral or solid name being enclosed in single quotes. This is followed by the molar volume of the mineral in units of cm3/mole.

<'MineralName' > <Molar Volume> <Number of species in reaction> <Stoichiometric coefficient> <'SpeciesName'> <Stoichiometric coefficient> <'SpeciesName'> …<Log K array> <Molecular Weight>

In this format, the number of pairs of stoichiometric coefficients and species names is determined by the preceding value in "Number of species in reaction" (if there is a mismatch, a read error will result). The length of the "Log K" array is given by the number following 'temperature points' in the first line of the database file.

An example is given by the entries for quartz and calcite:

```
'Quartz'  22.6880  1  1.00 'SiO2(aq)'  -4.6319  -3.9993  -3.4734
-3.0782    -2.7191  -2.4378  -2.2057  -2.0168  60.0843

'Calcite'  36.9340  3  -1.00 'H+'  1.00 'Ca++'  1.00 'HCO3-'
2.2257   1.8487  1.3330  0.7743  0.0999  -0.5838  -1.3262  -
2.2154  100.0872
```

Kinetic Database Entries

The kinetic database entries include information on rate formulations and coefficients to be used in a simulation. Some of these values can be overwritten in the input file at run time. With the exception of the case when generic_rates is specified with a value in the **RUNTIME** block, however, there must be an entry in the database corresponding to the particular reaction, even if some of the values are not used. The beginning of the section on solid-liquid phase kinetics in the database is marked by the string:

```
Begin mineral kinetics
```

which is not enclosed in quotes. Each entry is delimited by a line beginning with a "+" which speeds up the database searching. This is followed by a string giving the name of the solid phase. Names given in the input file (described below) must match these names for a particular reaction to be loaded. This name must also match an entry in the thermodynamic part of the database, although this requirement may be relaxed in the future for the special cases of irreversible reactions where thermodynamics may be irrelevant. The code also allows for multiple parallel reactions (multiple reactions running in parallel which affect the same solid phase) through the database entry on the line following the solid phase name. This line begins with the string "label = ", and is to be followed with the label of the rate law. The following (third) line begins with the string "type = ", which may include currently up to five different types of rate law (see below). The fourth line begins with "rate (25C) = ". The fifth line gives the activation energy for the reaction. While there are different options after the fifth line depending on the type of reaction involved (see below), all of the solid-liquid phase kinetic entries have a common input format for the first five lines following the "+" separator:

```
+--------------------------------------------------
Calcite
label = <name>
type = <type of reaction>
```

```
rate(25C) = <rate at 25C in log (base 10) units>
activation = <activation energy (kcal/mole)>
```

The “label” option can be used to construct an overall rate of reaction that is the sum of a number of parallel reaction pathways. An example would to use several database entries for “Calcite” with differing labels and rate laws to capture both the pH independent and pH dependent rates. After discussing the remainder of the input for the solid-liquid phase kinetics, an example of multiple parallel reactions involving calcite will be given.

Several types of reactions may be specified. The five options currently available are:

```
type = tst
type = monod
type = irreversible
type = PrecipitationOnly
type = DissolutionOnly
```

and are described briefly below.

TST: These three different reaction types involve different kinds of input following the “rate(25C)” because of the difference in the rate formulations. The “tst” rate law is described in detail by Lasaga (1981; 1984) and by Aagaard and Helgeson (1981) and takes the form:

$$R = A_m k_m \exp\left[\frac{-E_a}{RT}\right]\prod a_i^n \left[1-\exp\left(m_2 g^{m_3}\right)\right]^{m_1}$$

$$g \equiv \frac{\Delta G}{RT} = \ln\left[\frac{Q}{K_{eq}}\right]$$

where A_m is the mineral surface area, k_m is the intrinsic rate constant in units of mol/m^2/s, E_a is the activation energy (kcal/mole), Q_m is the ion activity product for the mineral-water reaction, K_{eq} is the corresponding equilibrium constant, and $\prod a_i^n$ is a product representing the inhibition or catalysis of the reaction by various ions in solution raised to the power n. Note that this last term operates far from equilibrium, that is, when Q_m/K_{eq} is very small (i.e., the mineral is very undersaturated).

Following the specification of the activation energy is the specification of the dependences of the far from equilibrium rate on various species in solution:

```
dependence :    <species name> <exponent> <species name> <exponent> …
```

The dependence of the far from equilibrium rate is given by pairs of species names in free format followed by the appropriate exponent.

An example of a database input which provides both pH dependent and pH independent calcite dissolution rates is given below:

```
+-------------------------------------------------
Calcite
label = default
type  = tst
rate(25C) = -6.19
```

```
activation = 15.0  (kcal/mole)
dependence :
+-------------------------------------------------
Calcite
label = h+
type  = tst
rate(25C) = -3.00
activation = 15.0  (kcal/mole)
dependence :     H+  1.0
```

A general form for a dependence on reaction affinity (or Gibbs free energy) is given following that given in Burch et al (1993) and Hellmann and Tisserand (2006). The affinity dependence of the rate is defined with the parameters *m_1, m_2, and m_3*. A more familiar form would be:

$$R = \left[1 - \left(\frac{Q}{K_{eq}}\right)^{m_2}\right]^{m_1}.$$

The form in the database is

```
AffinityDependence = m1  m2  m3
```

As an example, the rate law for albite dissolution proposed by Hellmann and Tisserand (2006) can be implemented by specifying two parallel rate laws

```
Albite
    label = HellmannFFE
    type = tst
    rate (25C) = -11.9897
    activation = 0.00 (kcal/mol)
    dependence :
    AffinityDependence = 1.00   0.0000798   3.81
+-----------------------------------------------------------------
Albite
    label = HellmannCTE
    type = tst
    rate (25C) = -13.743
    activation = 0.00 (kcal/mol)
    dependence :
    AffinityDependence = 1.17   1.00   1.00
+-----------------------------------------------------------------
```

Currently, it is only possible to specify a dependence on reaction affinity in the database. Only those rate laws in which a dependence on reaction affinity occurs (TST, PrecipitationOnly, and DissolutionOnly) will be affected.

Monod: Monod reactions take a slightly different form than do TST-type reactions. Following the specification of the activation energy, the Monod option has a field for inputting various "Monod terms", that is, the dependence of the reaction rate on electron acceptors and/or electron donors. The specification of Monod terms takes the form:

monod_terms : <electron donor or acceptor> <half-saturation constant> <electron donor or acceptor> <half-saturation constant> …

The half-saturation constant is assumed to be in concentration units of moles/kg water. This in turn is followed by a field for the specification of inhibition terms (also in moles/kgw):

inhibition_terms : <inhibitor (species name)> <inhibition constant> <inhibitor (species name)> <inhibition constant> …

The Monod terms take the standard form:

$$R_m = k_{\max}\left(\frac{C_i}{C_i + K_{half}}\right)$$

with the quantities in parentheses representing the "Monod term". Multiple Monod term may be specified, although the most common approach is to use a dual Monod form in which a dependence on an electron acceptor and on an electron donor are provided. An example involving aerobic respiration is given by:

```
+----------------------------------------------------
CH2O
label = aerobic_respiration
type  = monod
rate(25C) = -9.699
activation = 0.0
monod_terms :  O2(aq)  15.0e-06
inhibition :
```

The inhibition terms take a similar (if inverted) hyperbolic mathematical form:

$$I_m = \frac{K_{in} + C_i}{K_{in}},$$

where K_{in} is the inhibition constant and C_i refers to the inhibiting species. An example involving denitrification is given in the following database input:

```
+----------------------------------------------------
CH2O
label = denitrification
type  = monod
rate(25C) = -10.00          !  Regnier and Steefel (1999)
activation = 0.0  (kcal/mole)
monod_terms :  NO3-  45.0e-06
inhibition :  O2(aq)  30.0e-06
```

Note that both of these forms are typical of those used for single as opposed to dual Monod formulations. In these cases, the organic carbon is considered to be in excess of any limiting concentration. At this point, all Monod-type reactions are assumed to be irreversible (that is, no use is made of the equilibrium constants).

Irreversible: Irreversible reactions are the simplest kind in that no use is made of the equilibrium constants. The reactions are assumed to take the form

$$R_m = A_m k_m \exp\left(\frac{-Ea}{RT}\right)\prod a_i^n,$$

and are therefore similar to the TST form except that a dependence on the saturation state is missing. Because of the similarity to the TST reactions, the input is also similar, for example:

```
+-----------------------------------------------------
Iron
label = tce
type  = irreversible
rate(25C) = -6.92
activation = 15.0  (kcal/mole)
dependence :   TCE(aq) 1.0
```

where the rate of dissolution of zero-valent iron is here considered to have a first-order dependence on the concentration of TCE in solution.

<u>PrecipitationOnly:</u>

Invoking this option in the database causes the code to calculate mineral precipitation only—dissolution is suppressed. An example is given by:

```
KaoliniteYang
  label = Yang
  type  = PrecipitationOnly
  rate(25C) = -13.47
  activation = 0.0  (kcal/mole)
  dependence :
  AffinityDependence = 1.00   2.07468   -1.00000
+----------------------------------------------------
```

<u>DissolutionOnly:</u>

Invoking this option in the database causes the code to calculate mineral dissolution only—precipitation is suppressed. An example is given by:

```
KaoliniteYang
  label = Yang
  type  = DissolutionOnly
  rate(25C) = -12.9393
  activation = 0.0  (kcal/mole)
  dependence :
  AffinityDependence = 1.00   0.5   1.00
+----------------------------------------------------
```

This option is useful for specifying different rate laws for precipitation and dissolution where both PrecipitationOnly and DissolutionOnly are invoked separately for the same mineral.

Entries in Input File

The simplest possible form for specification of a solid-aqueous phase reaction in the database is to give the solid or mineral name on a separate line within the **MINERALS** keyword block without any accompanying information. If a database sweep has been specified previously, then the list of minerals may be copied from the file "'input file name prefix.out" to which they have been written and then pasted directly into the **MINERALS** keyword block. However, unlike similar operations for secondary species and gases, the minerals specified in the input file also require kinetic information which resides in a separate part of the database from the thermodynamic entries for minerals or solid phases. Since the kinetic database is not as complete as the thermodynamic database (letters written to funding programs within DOE lamenting this situation will be appreciated), some minerals found during the sweep may not have kinetic database entries yet. This will be remedied in future releases by 1) augmenting the kinetic database for CrunchFlow and 2) by sweeping the kinetic database in addition to the thermodynamic database for solid phase reactions which can be loaded directly as kinetic reactions. The database sweep will tend to find a relatively large number of phases which may or may not actually form at the temperature of interest. It is common, for example, to find various forms of garnet supersaturated at 25C—geochemists and petrologists know that these phases precipitate only at high temperature. Thus, editing of the mineral list generated by a database sweep is essential.

Without any additional information, the code will assume that the "default" reaction is to be loaded if only the name of the solid phase is given, that is, it will search for a kinetic entry of that name labeled as "default".

Following the specification of the reaction, the mineral kinetic options that can be specified in the input file include:

```
-label <label>
-rate <rate>
-activation <activation energy>
-suppress_precipitation  <ε>
-local_equilibrium
-associate <mineral_name>
```

To load solid-liquid phase reactions with labels other than "default", one uses the form:

```
<Reaction name> -label <label>
```

where "label" is a string which must match one in the kinetic database. Similarly, one can overwrite the reaction rate in the database with a specification in the input file:

```
<Reaction name> -label  <label>   -rate <rate>
```

where the rate is assumed to be the logarithm of the rate in units of moles/m^2/s at 25C. The activation energy may also be overwritten:

```
<Reaction name> -label  <label>   -activation <activation
energy>,
```

where the activation energy is in units of kcal/mol.

Previously, it was possible to specify a threshold for nucleation, but implemented in the simple fashion in which it was, numerical convergence was poor due to the discontinuity in rates across the nucleation threshold. The Newton method for nonlinear equations performs poorly with functions that are not continuously differentiable. Now, one can achieve at least suppression of precipitation by using the *–suppress_precipitation* option.

```
<Reaction name> -suppress_precipitation  <ε>,
```

where ε is the value of the absolute value of the undersaturation expressed as log(Q/Keq) where the dissolution rates = ½ of its maximum value. However, normally it is preferred to use the DissolutionOnly in a kinetic entry in the database.

For the case of linear kinetics, the code uses an analytical expression for the approach to equilibrium that is continuously differentiable, with the rate given by:

$$R_d = 0.5 - 0.75\xi + 0.25\frac{\xi^3}{\varepsilon^3},$$

where the quantity ξ is defined by

$$\xi = Log\left(\frac{Q}{K_{eq}}\right) + \varepsilon \ .$$

For nonlinear kinetics, the factor of ε is not used and precipitation is simply suppressed, while the rate law provided in the database is used to give the dependence on Gibbs free energy.

Local equilibrium with respect to mineral phases (variously known as the *local equilibrium approximation or fantasy*, depending on one's point of view) can be specified through a combination of parameters. To suppress the reduction in reactive surface area as the mineral volume fraction of $\rightarrow$ 0, the following option may be specified

```
<Reaction name> -local_equilibrium
```

In other words, the initial reactive surface area remains fixed even during dissolution as long as the mineral is present at all. This is of course unrealistic, but the local equilibrium fantasy has been so drummed into a generation of geochemists and petrologists that it is impossible to dispense with this option. This alone does not enforce local equilibrium, but when combined with a sufficiently large rate constant and reactive surface area, equilibration within about 10^{-4} log(Q/K_{eq}) units of equilibrium is possible. The rate constant and/or reactive surface area should not be set arbitrarily high, however, since this leads to poor numerical convergence due to the stiffness of the governing equations. A log rate constant in the range of -3.00 to -5.00 should provide local equilibrium behavior, although this depends in detail on the magnitude of the transport rates. The following example should provide local equilibrium with respect to the minerals *portlandite* and *Ca-oxalate* even under relatively high flow rates

```
MINERALS
Portlandite       -rate -3.00  -local_equilibrium
Ca-Oxalate        -rate -3.00  -local_equilibrium
END
```

Another option that has been recently added is given by

```
 <Reaction name> -associate <mineral_name>
```

which is used to associate more than one reaction stoichiometry and equilibrium constant with the same mineral. This is relevant normally when aqueous kinetics are involved and the reaction for the same mineral is written as a kinetic reaction involving different sets of primary species. The most familiar example would be the oxidation of organic carbon by various electron acceptors, none of which are in equilibrium with each other (and therefore they are listed as Primary Species—see below). One example is given by the reductive dissolution of amorphous iron hydroxide by two different mechanisms: 1) the reductive dissolution mediated by the bacteria *Geobacter,* and 2) the abiotic reduction by hydrogen sulfide:

$$\begin{aligned}&\mathrm{FeOOH(s)} + 1.925\mathrm{H}^+ + 0.033\mathrm{NH}_4^+ + 0.208\mathrm{CH_3COO}^- \rightarrow \\ &\mathrm{Fe}^{2+} + 0.033\mathrm{C_5H_7O_2N_{(Fe)}} + 0.25\mathrm{HCO}_3^- + 1.6\mathrm{H_2O}\end{aligned}$$

designated in the input file and database as "Iron_DIRB", while the reaction

$$2FeOOH + HS^- + 5H^+ \rightarrow 2Fe^{2+} + S_0 + 4H_2O$$

could be designated as `Iron_H2S.` The code thinks these are two separate phases because of the different entries and stoichiometries, unless the "associate" keyword is used. If one mineral is associated with another, then only the volume percentage and reactive surface of the "associated" mineral is updated. So in the example above, the following usage:

```
MINERALS
Iron_DIRB
Iron_H2S        -associate Iron_DIRB
END
```

would result in the dissolution according to the `Iron_H2S` kinetic pathway affecting the volume fraction and surface area of the `Iron_DIRB` phase. To see that this is working, users can check to see which of these phases is being updated as a function of time in the volume#.out files.

AQUEOUS_KINETICS

The keyword block and database entries for aqueous phase (homogeneous) kinetics currently have a simple form which will be changed in later releases to conform more closely to the format used for mineral-water (or solid-liquid) reactions. Aqueous phase reactions must be made up entirely of primary species—the concept of a "secondary species" in CrunchFlow is restricted to reactions where an equilibrium relationship between the species in the reaction applies. In an aqueous phase kinetic reaction, species which might have been secondary species if equilibrium applied need to be moved to the list of primary species or the reaction will not be loaded. Consider the case of the Fe(II)-Fe(III) redox couple. In the case where the oxidation of Fe(II) by molecular oxygen occurs rapidly and therefore can be assumed to be at equilibrium, one could use a list of primary species given by the following to describe the system:

```
PRIMARY_SPECIES
H+
Fe++
O2(aq)
END
```

followed by specification of Fe+++ in the list of secondary species. This would then load the reaction (neglecting the trailing equilibrium constants):

```
'Fe+++'  4  -0.50 'H2O'  0.25 'O2(aq)'  1.00 'Fe++'  1.00 'H+'
```

along with its associated equilibrium constants. Fe^{+++} would then become a secondary species contributing to the total concentrations of all of the primary species in the reaction (not only Fe^{++}, but also O_2(aq) and H^+). A fuller discussion of this can be found in Steefel and MacQuarrie (1996) or in Bethke (1996). However, if one wanted to treat this reaction as a kinetic one, then Fe^{+++} needs to be listed as a primary species along with the others:

```
PRIMARY_SPECIES
H+
Fe++
O2(aq)
Fe+++
END
```

If this is done, then the reaction involving Fe^{+++} given above will not be loaded from the thermodynamic portion of the database. Left like this, Fe^{++} and Fe^{+++} would be treated as if they were separate elements with no reaction relationship linking their concentrations. This is the so-called "redox disequilibrium" option found in a number of the reaction path codes. In **CrunchFlow**, however, one can add a kinetic reaction linking these species using the **AQUEOUS_KINETICS** keyword block.

Database Formats

Unfortunately the aqueous kinetics inputs have not yet been shifted over to the more readable format of the mineral kinetics section. In addition, the aqueous kinetic reactions do not currently make use of the equilibrium constants and reaction stoichiometries found in the thermodynamic portions of the database. This is nominally because kinetic pathways may have a specific stoichiometry which differs from that found in the thermodynamic entries. As in the case of mineral-water reactions, parallel reaction pathways may be operating, each associated with its own different reaction stoichiometry. In the initialization/speciation procedure, since equilibrium is assumed, only a single reaction stoichiometry is needed (Steefel, 2000). Any reaction stoichiometry in terms of the primary species can be used and will give identical results mathematically. The aqueous kinetics section follows the MINERALS section of the database and is begun with the keyword phrase:

```
      Begin aqueous kinetics
```

and is terminated with the keyword phrase:

```
      End of aqueous kinetics
```

Entries in the aqueous kinetics section of the database take the form

<Aqueous reaction label> <Number of parallel rate laws> <'ReactionType'> <Number of species in reaction (integer)> <Stoichiometric coefficient> <'SpeciesName'> <Stoichiometric coefficient> <'SpeciesName'> … <Log K>

which is then followed by the number of lines corresponding to the number of parallel rate laws which are described below.

A minimum of one rate law needs to be given in the database following the kinetic reaction stoichiometry. "Reaction type" specified in the first line refers to rate laws of the type "tst", "monod", or "irreversible" just as in the solid-liquid phase reaction input (although the format of the input clearly differs). A fuller description of the various reaction "types" is given below.

Types of Rate Laws

TST: The "tst" rate law is described in detail by Lasaga (1981; 1984) and by Aagaard and Helgeson (1981) and takes the form:

$$R_s = k_s\left(1 - Q_s/K_{eq}\right)\prod a_i^n \ ,$$

where k_s is the intrinsic rate constant in units of mol/kg water/yr, Q_s is the ion activity product for the mineral-water reaction, K_{eq} is the corresponding equilibrium constant, and $\prod a_i^n$ is a product representing the inhibition or catalysis of the reaction by various ions in solution raised to the power n. Unlike the database entries for solid-liquid phase reactions, there is no dependence on surface area and currently no allowance for a temperature dependence. Rate laws for "tst"-type reactions take the following form:

<rate (mol/kgw/yr)> <Number of species dependences> <’SpeciesName’> <Exponent> <’SpeciesName’> <Exponent> …

As an example, consider the kinetically controlled oxidation of Fe++ by molecular oxygen which uses two parallel reactions (one pH dependent and one pH independent) following the rate law proposed in Singer and Stumm (1970):

```
FeII_oxidation 2 ‘tst’ 5 1.0 ‘Fe+++’  0.500 ‘H2O’ –0.25 ‘O2(aq)’ 
–1.0 ‘Fe++’ –1.0 ‘H+’ 8.4887

   1.53e-06  3 ‘tot_Fe++’ 1.0  ‘O2(aq)’ 1.0 ‘H+’ –2.0

   41.4848  2 ‘tot_Fe++’ 1.0  ‘O2(aq)’ 1.0
```

As in the case of solid-liquid phase reactions, the reaction rate can be made to depend on the total concentration of a species (Fe++ in this case) by prepending the string “tot_” to the species name.

Monod: Monod reactions take a slightly different form than do TST-type reactions. The Monod option has a field for inputting various “Monod terms”, that is, the dependence of the reaction rate on electron acceptors and/or electron donors:

$$R_m = k_{\max}\left(\frac{C_i}{C_i + K_{half}}\right)$$

where C_i is the concentration of the electron acceptor or donor, K_{half} is the half-saturation constant in moles/kg water and k_{max} is the maximum rate of the reaction (when the concentration of the electron donor or acceptor >> the half-saturation constant)

The specification of Monod terms takes the form:

<maximum rate> <number of Monod terms (integer)> <electron acceptor or donor> <half-saturation constant> <electron acceptor or donor> <half-saturation constant> …

where the “electron donor or acceptor” is a species name which again can be prepended with the string “tot_” to designate a total concentration.and the half-saturation constant is the concentration of the electron acceptor or donor at which the rate is ½ of its maximum value.

Inhibition terms of the following form may also be included:

$$I_m = \frac{K_{in} + C_i}{K_{in}},$$

where K_{in} is the inhibition constant and C_i is the concentration of the electron acceptor or donor. Inhibition terms in the aqueous kinetics section are specified by the string ‘Inhibition’ enclosed in single quotes followed on the same line by:

<Number of inhibiting species> <SpeciesName> <Inhibition constant> <SpeciesName> <Inhibition constant> …

Irreversible: Irreversible reactions are the simplest kind in that no use is made of the equilibrium constants. The reactions are assumed to take the form

$$R_s = k_s \prod a_i^n ,$$

and are therefore similar to the TST form except that a dependence on the saturation state is missing. Such a reaction rate law can be used to an exponential decay reaction. In combination with a series of other decay reactions, it can be used to describe an entire decay chain. The decay chain ^{241}Pu $\rightarrow$ Am $\rightarrow$ Np can be represented by the following entries in the database:

```
Pu++++241_decay   1  'irreversible'   2  -1.0  'Pu++++241' 1.0 'Am+++'
    4.83E-2  1  'tot_Pu++++241' 1.0
Am+++_decay   1  'irreversible'   2  -1.0  'Am+++' 1.0 'Np++++'
     1.60E-3  1  'tot_Am+++' 1.0
```

Since the reactions are designated as irreversible, no equilibrium constants are needed.

Entries in Input File

Aqueous kinetic reactions are entered in the input file using the reaction label that in each case begins a database entry. So for the example above, the **AQUEOUS_KINETICS** keyword block would look like the following:

```
AQUEOUS_KINETICS
Am+++_decay
Pu++++241_decay
END
```

No other options are currently allowed in the input file.

ION_EXCHANGE

An ion exchange reaction can be described via a mass action expression with an associated equilibrium constant (Vanselow, 1932; Sposito, 1981; Appelo and Postma, 1993). The exchange reaction can be written in generic form as

$$vACl_u(aq) + uBX_v(s) \leftrightarrow uBCl_v(aq) + vAX_u(s)$$

where X refers to the exchange site occupied by the cations A^{u+} and B^{v+}. The equilibrium constant, K_{eq}, for this reaction can be written as (Vanselow, 1932)

$$K_{eq} = \frac{(BCl_v)^u (AX_u)^v}{(ACl_u)^v (BX_v)^u}$$

where the parentheses () refer to the thermodynamic activities. Several activity conventions are in wide use. One possibility is the Gaines-Thomas activity convention, which assumes a reaction stoichiometry of the following form (Appelo and Postma,1993), written here assuming the Cs^+ is the relevant cation of interest

$$Cs^+ + \frac{1}{m} MX(i)_m \leftrightarrow CsX(i) + \frac{1}{m} M^{m+}$$

where M is the competing cation (Na^+, K^+, Ca^{++}), m is its charge, and *X(i)* refers to the *i*th type of exchange site. In the Gaines-Thomas convention, each exchange site, *X(i)*, has a charge of -1. The activities of adsorbed species correspond to the charge equivalent fractions, $\beta(i)_M$,

$$\beta(i)_M = \frac{z_M q(i)_M}{\sum_M z_M q(i)_M} = [X(i)_M]$$

where z_M is the charge of cation M, $q(i)_M$ is the concentration of adsorbed cation M in exchange site i (moles/g), and the square brackets denote activities. The Gapon activity convention is obtained by writing the reactions in every case with a single exchanger (Appelo and Postma, 1993). Alternatively, the Vanselow convention (Vanselow, 1932) describes the exchanger activity with mole fractions

$$\beta(i)_M = \frac{q(i)_M}{\sum_M q(i)_M} = [X(i)_M].$$

The exchange reactions can then be used to write a mass action equation for binary Cs-M exchange:

$$K_{M/Cs} = \frac{\beta(i)_M{}^{1/m}[Cs^+]}{\beta(i)_{Cs}[M^{m+}]^{1/m}} = \frac{[X(i)_M]^{1/m}[Cs^+]}{[X(i)_{Cs}][M^{m+}]^{1/m}}$$

In a single-site ion exchange model, the CEC is equal to the sum of the charge equivalent concentrations of the adsorbed cations:

$$CEC = \sum_M z_M q_M$$

while in a multi-site model, the CEC is the charge summed over all of the cation exchange sites (Cernik et al., 1996; Voegelin et al., 2000)

$$CEC = \sum_i \sum_M z_M q(i)_M$$

In CrunchFlow, exchangers may be specified with the format

```
    exchange   exchanger_name
```

where *exchanger_name* is the name of the exchanger given in the database. Multiple exchangers can be listed. If the exchanger name is not followed by anything, it is assumed that exchange is on the bulk material, in which case a CEC is calculated from the combination of the solid phase density, porosity, and liquid saturation (see Section 1.2.14). Alternatively, it is possible to specify exchange on a specific mineral, in which case the CEC is calculated from the volume fraction of the mineral, which may change with time, for example:

```
    exchange   Xkaol-  on Kaolinite
```

Also required is the keyword *convention*

```
    convention   activity_convention
```

where activity_convention must be either *Gaines-Thomas* (of which Gapon is a variant) or *Vanselow*.

SURFACE_COMPLEXATION

Surface complexation follows the approach outlined in Dzombak and Morel (1990), with either a double layer or non-electrostatic model possible. Currently, complexation must be on a specific mineral, so a valid mineral name (listed in the **MINERALS** keyword block) must be given, for example:

```
    >FeOH_strong  on  Fe(OH)3
    >FeOH_weak    on  Fe(OH)3
```

To specify a non-electrostatic model, the mineral name should be followed by the keyword *–no_edl*, for example:

```
    >FeOH_strong  on  Fe(OH)3  -no_edl
    >FeOH_weak    on  Fe(OH)3  -no_edl
```

The capability for surface complexation on the bulk material will be added soon.

GEOCHEMICAL CONDITIONS

Geochemical conditions are keyword blocks which may occur more than once and are used to set up boundary and initial conditions and source terms. In each case, they provide information on the actual concentrations of primary species and minerals needed to carry out a distribution of species calculation.

Only after a particular geochemical condition is read and "processed" is it allocated as a particular boundary or initial condition or as a source term.

Geochemical conditions are input via a CONDITION keyword block. The CONDITION string is followed in each case by a label which identifies the condition elsewhere in the input file. Most of the information in a condition block pertains to aqueous and solid phase concentrations—however, certain parameters may be input which apply to the condition as a whole. These are discussed in the following section.

Units

Concentration units may be set using the units keyword within a particular condition. The change in concentration units only applies to the condition within which it is set. The format for input of different units is:

units <concentration units>

The default concentration units for aqueous species is molality (moles solute per kg water). The additional options include:

Concentration Units	**Keyword Designator**
moles/kg water	mol/kg
millimoles/kg water	mmol/kg
micromoles/kg water	umol/kg, micromol/kg
parts per million	ppm

Table 3: Options for specifying concentrations in geochemical conditions

In each case, the use of the "ppm" units refers to the parts per million of the solute using the molecular weight.of the primary species to which it refers. Therefore, if the primary species is NO_3-, then the concentration in ppm is calculated using the molecular weight of NO_3- (not, for example, using N).

Equilibrate_surface

When specified on a separate line, this keyword instructs the code to partition all of the total concentrations between the aqueous and solid phases (i.e., surface hydroxyl sites and exchangers). It does not partition any mass through precipitation-dissolution reaction. This instruction will only apply to species for which a “total concentration” constraint is used (described below). This instruction can also be specified on a species by species basis as discussed below. If the equilibrate_surface instruction is absent, then the code assumes that the “total concentration” refers only to the aqueous phase and the exchanger and surface hydroxyl concentrations are calculated accordingly.

Temperature

Keyword used to set a temperature within an individual geochemical condition.

Syntax*:* **temperature** *value*

Default: *Provided by value in TEMPERATURE keyword block*

Explanation: This keyword overrides the value set in the **TEMPERATURE** keyword block. If absent, that value is used for the geochemical condition.

Set_porosity

Keyword used to set a porosity within an individual geochemical condition.

Syntax*:* **set_porosity** *value*

Default: *Provided by value in POROSITY keyword block*

Explanation: This keyword overrides the value set in the **POROSITY** keyword block. If absent, that value is used for the geochemical condition.

NOTE: This option only works if ***fix_porosity*** *is set in the* ***POROSITY*** *keyword block. Otherwise, porosity is calculated from the 1 minus the sum of the volume fractions of the minerals.*

Set_saturation

Keyword used to set a liquid saturation within an individual geochemical condition.

Syntax*:* **set_saturation** *value*

Default: *Provided by value in RUNTIME keyword block, otherwise 1.0*

Explanation: This keyword overrides the value set in the **RUNTIME** keyword block. If absent, that value is used for the geochemical condition.

Types of Constraints: Aqueous Species

Various types of constraints may be specified for aqueous species in the distribution of species calculations carried out on each of the geochemical conditions. These constraints are summarized in Table 4.

Constraint Type	**Use**	**Example**

Total Concentration	Mole balance on total aqueous or total aqueous+sorbed concentrations	Na+ 0.001
Species Concentration	Specify an individual primary species concentration	Na+ 0.001 species
Species Activity	Specify an individual primary species activity	Na+ 0.001 activity
pH	Specify activity of hydrogen ion	pH 7.8
Gas	Equilibrate primary species with a gas	O2(aq) O2(g) 0.20
Mineral	Equilibrate primary species with a mineral	O2(aq) Pyrite
Charge	Charge balance the solution using the primary species	Na+ charge

Table 4: Constraints available for specifying aqueous species concentrations.

The most commonly used constraint is the "total concentration" constraint. Analyses of natural waters generally report total concentrations rather than individual species concentrations. Or someone may know how much total mass of a chemical element they added to an experimental system. While specification of individual species concentrations is relatively uncommon, specification of species activity is common, especially for hydrogen ion (pH) and for molecular oxygen (O_2(aq)).

Total Concentration Constraint

The total concentration constraint signals to CrunchFlow to carry out a total mole balance calculation. If the balance is over the aqueous phase only (the default), the equation takes the form:

$$T_j = C_j + \sum_{i=1}^{Ns} \nu_{ij} C_i$$

where *Tj* is the total concentration of the primary species, *Cj* is the individual primary species concentration, ν_{ij} is the stoichiometric coefficient, and *Ci* is the concentration of the *Ns* secondary species. If the keyword *equilibrate_surface* is specified either as a global parameter within a condition or for an individual primary species, then the mole balance equation also includes the mass of the primary species incorporated in ion exchange sites and as surface complexes:

$$T_j = C_j + \sum_{i=1}^{Ns} \nu_{ij} C_i + \sum_{k=1}^{Nexs} \nu_{kj} C_k + \sum_{l=1}^{Nsurf} \nu_{lj} C_l \ ,$$

where *Ck* refers to the *Nex* exchange species and *Cl* refers to the *Nsurf* surface complexes. To specify a total concentration within a condition, one uses the form:

<PrimarySpeciesName> <Concentration>

or equivalently,

<PrimarySpeciesName> <Concentration> total

where the “total” is assumed to be the default. To distribute the specified mass over the exchangers and surface hydroxyl sites as well, the user follows the concentration value with the keyword *equilibrate_surface* as in the first three entries of the following example:

```
Am+++           1.00e-10  equilibrate_surface
Np++++          1.00e-10  equilibrate_surface
Pu++++241       1.00e-10  equilibrate_surface
K+              8.70e-5
Mg++            2.06e-5
```

In this example, the Am, Np, and Pu will be partitioned between the aqueous phase and the exchange and surface hydroxyl sites (if present), while the total concentration in the case of K and Mg will be distributed completely within the aqueous phase.

Species Concentration Constraint

The species concentration constraint is straightforward and need not be discussed here other than to show how it is specified in the input file. Since the total concentration constraint is considered the default, it is necessary to specify the keyword species after the concentration value:

<PrimarySpeciesName> <Concentration> species

Species Activity Constraint

Specifying a species activity is also straightforward and takes the form:

<PrimarySpeciesName> <Concentration> activity

A specification of the pH is a special case which does not require the trailing *activity* keyword

```
pH      7.8
```

Gas Constraint

The gas constraint is used to calculate a primary species activity so as to obtain equilibrium with respect to a gas at a specified partial pressure/fugacity. At this point, no fugacity corrections are available in the code, so effectively partial pressure = fugacity. Partial pressure is input in bars. The format for using the gas constraint is to follow the primary species name by the name of the gas with which the species is to be equilibrated and then the partial pressure in bars:

<primary species name> <gas name> <partial pressure (bars)>

To adjust the O2(aq) activity to be at equilibrium with respect to atmospheric oxygen, one would use the following form:

```
O2(aq)    O2(g)   0.20
```

Mineral Constraint

The mineral constraint operates in a similar fashion to the gas constraint except that no trailing value is needed. This is because minerals (or solid phases) are assumed at this stage to be pure, that is, their activities = 1. Unlike some other codes like PHREEQC, CrunchFlow does not carry out a formal reaction path calculation to attain equilibrium with respect to the mineral. The mineral constraint is used together with the other constraints in the geochemical system to provide the same number of constraint equations as there are unknown and this system of equations is solved without any path dependence. The format for a mineral constraint is simply:

<primary species name> <mineral name>

Charge Constraint

The charge constraint may be somewhat more difficult to implement because the user may choose to balance on an anion when the remainder of the solution is already negatively charged or vice versa. In this case (without the proper safeguards), the concentration of the charge-balancing species would go to 0. If this situation arises, an error message will result and the run will terminate. The format for specifying a charge balance is:

<primary species name> charge

Solid Phase Inputs

In most cases, information on mineral concentrations (or volume fractions) and mineral surface areas are needed to completely describe a geochemical condition. Those cases which don't require such information are typically boundary conditions or source terms where only the aqueous phase is entering the domain. In this case it is possible to leave the mineral input out of the condition.

The code does not check to see whether mineral input is or is not needed—if it doesn't find any information on a mineral or solid phase which has been loaded in the **MINERALS** keyword block, it will assume its volume fraction = 0 and use a default bulk surface area of 100 m^2 mineral/m^3 porous medium. In this way, a mineral which is neglected in a particular condition is treated as a "potential secondary phase", that is, it is not initially present but it could form if supersaturated. Since boundary conditions and source terms represent "ghost cells or zones" outside of the domain which are therefore not computed as a function of time, supersaturation of a mineral phase will have no effect.

Mineral Volume Fractions

The first values to be input following a mineral/solid phase name is the volume fraction. The ability to add mineral concentrations in other units (mass fraction, etc.) will be added in future releases. The format is:

<name of solid phase> <volume fraction>

Surface area information is appended to the same line and is described below.

Mineral Surface Area Options

Mineral surface area is important in two ways: 1) it gives the "reactive surface area" to be used in mineral dissolution and precipitation reactions, and 2) it provides the surface area upon which adsorption (surface complexation) may occur (see section on Surface Complexation Input). There are currently two options for specifying mineral surface area: 1) *bulk surface area* in units of m^2 solid phase/m^3 porous medium, or 2) *specific surface area* in units of m^2/g. The format is:

<name of solid phase> <volume fraction> <surface area option> <value>

where the surface area options are *bulk_surface_area* or *bsa* and *specific_surface_area* or *ssa* respectively for the bulk and specific surface area options. If the surface area option is left off, the code assumes by default that the trailing numerical value refers to the bulk surface area.

Specification of a bulk surface area will lead to a calculation of specific surface area according to:

$$A_{specific} = {A_{bulk} V_m}\Big/{\phi_m MW_m},$$

where V_m is the molar volume of the solid phase, ϕ_m is the initial volume fraction, and MW_m is the molecular weight of the phase. The specific surface area is then used together with other surface complexation parameters to calculate concentrations of surface hydroxyl sites per volume porous medium.

The two surface area options lead to different methods for computing mineral surface area as a function of time.

<u>Bulk surface area:</u> In the case where the bulk surface area is specified, the reactive surface area of a solid phase as a function of time is calculated according to:

$$A_{bulk} = A_{bulk}^{initial} \left(\phi_m \Big/ \phi_m^{initial} \right)^{2/3} \left(\phi \Big/ \phi^{initial} \right)^{2/3} \quad \text{(dissolution)},$$

$$A_{bulk} = A_{bulk}^{initial} \left(\phi \Big/ \phi^{initial} \right)^{2/3} \quad \text{(precipitation)},$$

where ϕ refers to the porosity and ϕ_m refers to the individual mineral volume fraction. The inclusion of a 2/3 dependence on porosity is chiefly to ensure that as the porosity goes to 0, so too does the mineral surface area available for reaction. This formulation is used primarily for primary minerals (that is, minerals with initial volume fractions > 0). For secondary minerals which precipitate, the value of the *initial* bulk surface area specified is used as long as precipitation occurs—if this phase later dissolves, the above formulation is used, but with an arbitrary "initial volume fraction" of 0.01. Clearly the expressions above are only a very approximate means of calculating the reduction in surface area of a secondary phase if later dissolution occurs. More accurate for secondary minerals (and therefore recommended) is the use of specific surface area.

<u>Specific surface area:</u> When specific surface area is selected in the input file, the reactive surface area used for mineral precipitation and dissolution is calculated from:

$$A_{bulk} = {\phi_m A_{specific} MW_m}\Big/{V_m}.$$

This is the preferred approach for precipitation of secondary phases in particular, since the bulk surface area (BSA) approach currently uses the initial surface area only (i.e., the reactive surface area of a precipitating phase does not change as the secondary mineral concentration evolves). In contrast, with the specific surface area option, evolution of the mineral volume fraction causes the bulk surface area to evolve according to the equation above.

If the initial volume fraction of the solid phase is > 0, then the $A_{bulk}^{initial}$ is calculated directly from the specific surface area. If the initial volume fraction = 0 (that is, we are dealing with a secondary mineral initially not present at all), the specific surface area cannot be evaluated without some other procedure. To handle this case of a secondary phase not initially present, one can specify a "threshold mineral volume fraction", which is used to calculate the bulk surface area until the computed time-evolving volume fraction exceeds the threshold value. A threshold value for the case of an initial volume fraction = 0 is set by

```
<name of solid phase>  0.0  specific_surface_area <value> <threshold volume fraction>.
```

So, for example, we could specify that the bulk surface area of calcite (which is initially not present) be calculated from a threshold volume fraction of 0.0001 and a specific surface area of 2 m^2/g with the following form:

```
Calcite  0.0  specific_surface_area  2.0   0.0001
```

With this formulation, the bulk surface area of calcite would be calculated directly from the calcite volume fraction and specific surface area once it exceeded the threshold value of 0.0001 set her. This is a simple if not particularly mechanistic way to capture a nucleation event.

Surface Hydroxyl Concentrations

If surface complexation is included in a particular problem, then surface hydroxyl concentrations must be specified in each of the geochemical conditions—there is currently no default value available. The information which needs to be provided is the surface hydroxyl site density, ρ_{sites} , in units of moles sites per m^2 mineral. This information is combined with the specific surface area, $A_{specific}$, to calculate a concentration of surface hydroxyl sites per m^3 porous medium:

$$C_{sites} = \frac{\rho_{sites} A_{specific} MW_m \phi_m}{V_m}.$$

The format for inputting the molar density of sites is (mol/m^2):

```
<Surface complex>   Value
```

Exchanger Concentrations

The concentration of ions on exchangers can be set in several ways, with the most important distinction being between 1) exchange on the bulk material, and 2) exchange on specific minerals. In the case of exchange on a specific mineral, the code will link the concentration of exchange sites (i.e., the cation exchange capacity, or CEC) to the concentration of that mineral. Thus, an increase in the volume fraction of a secondary phase (e.g., clay as a result of chemical weathering or Fe-oxyhydroxide as a result of corrosion) will increase the cation exchange capacity of the bulk material.

Exchange on bulk material: This option is used where no specific mineral has been identified as the exchanger phase using the *on mineral* syntax in the **ION_EXCHANGE** keyword block. To calculate total concentration of exchange sites per bulk porous medium, the cation exchange capacity represented by that site needs to be specified along with the density and the total volume fraction of the bulk solid phase. This information must be provided in each geochemical

To specify the cation exchange capacity associated with each exchanger in a geochemical condition, the following syntax is used:

```
        Exchanger     -cec    Value
```

where the CEC is given in units of equivalents per gram solid. The name of the exchanger must correspond to one of those identified in the **ION_EXCHANGE** keyword block.

Solid density may be specified one of three ways:

1. Direct specification of the bulk solid density (kg/m^3)

```
SolidDensity    Value
```

2. Specification of the solid:solution ratio (g/kg water ~ g/L):

```
SolidDensity    -ss_ratio   Value
```

3. Calculation based on the sum of the mineral volume fractions:

```
SolidDensity   CalculateFromMinerals
```

Exchange on a specfic mineral: Here, the total number of equivalents of exchange sites is calculated from the combination of the mineral volume fraction and the cation exchange capacity per gram of that mineral. The solid density, therefore, is not relevant (the keyword is therefore ignored), although the values of the porosity and liquid saturation affect the result (see below). Mineral densities are calculated from the combination of molar volumes and molecular weights in the database. As in the case of exchange on the bulk material, the CEC is specified with a similar format, but here the units are equivalents per gram mineral:

```
Exchanger     -cec    Value
```

INITIAL_CONDITIONS

In order to run a reactive transport or a reaction path calculations, initial conditions must be set. Even in the case where the desired result is a steady state simulation in which initial conditions are irrelevant, the code must still achieve this steady state by stepping through time. Only with the *speciate_only* or *database*_sweep keywords set to true in the **RUNTIME** keyword block is it possible to proceed without specifying initial conditions for the entire spatial domain.

Required keyword block followed by a specification of the name of the geochemical condition to be used, followed in turn by the grid cells where the initial condition is to be set. The initial condition can be fixed for the entire course of the simulation using the optional appended keyword *fix*.

Syntax*: condition_name JX-JX [JY-JY] [JZ-JZ] [fix]*

```
Default:  None.
```

Explanation: Using this keyword block, geochemical conditions are distributed over the spatial domain. Multiple specifications can occur. Initial conditions listed later in the input file will overwrite conditions specified above. The optional parameter fix can be appended to the output so as to fix this geochemical condition for the entire course of the simulation. The code now assumes that if fix appears, it will occur immediately after the specification of the X coordinates (JX) in a 1D problem, immediately after the Y coordinates (JY) in a 2D problem, and after the Z coordinate (JZ) in the case of a 3D problem. In the example below, the code will verify that the geochemical condition names (quartz_zone, portlandite_zone, and nonreactive_zone) have been provided in the input file. This keyword block, therefore, is read after the geochemical conditions specified in the input file are read. The code will also verify that every grid cell in the domain is specified, implying that this block is read after the **DISCRETIZATION** keyword block.

Example:

```
INITIAL_CONDITIONS
quartz_zone        1-42    1-42
portlandite_zone   22-33   1-42
nonreactive_zone   42-42   42-42  fix
END
```

BOUNDARY_CONDITIONS

Boundary conditions are required for any problem involving transport. The input currently involves specifying boundary conditions, corresponding essentially to "ghost cells" outside of the domain, at the

Syntax*:*

```
x_begin  condition_name  boundary_condition_type
x_end  condition_name  boundary_condition_type
[y_begin  condition_name  boundary_condition_type]
[y_end  condition_name  boundary_condition_type]
[z_begin  condition_name  boundary_condition_type]
[z_end  condition_name  boundary_condition_type]
```

Default: None.

Explanation: Boundary conditions are set for ghost cells outside the faces of the 3D domain. Currently, uniform boundary conditions are specified over the entire face, although this is being revised to provide more flexibility. For additional flexibility, the user may fix initial conditions at the first or last grid cell in a coordinate direction, thus making it possible to specify multiple geochemical conditions on any one face. For fixed flux problems, it may be easier to make an injection well with the correct flux at the grid cell along the boundary. Boundary condition types are *Dirichlet* (used only in GIMRT mode) and *flux.* In the case of a *flux* specification, the code uses the value and direction of the advective flux that is provided—if the flux is into the domain, then the upstream ghost cell will be used, if out of the domain, then there is no effect. For a pure diffusion problem (i.e., with no advective flux via flow across the boundary of the domain), a *flux* specification will result in a no-flux boundary. In the case of a *Dirichlet* boundary in GIMRT mode, this will cause both advective and diffusive flux to be calculated. To use a Dirichlet condition in OS3D mode requires that an initial condition be fixed just inside the boundary of the problem.

TRANSPORT

The TRANSPORT keyword block reads in the inputs controlling transport of species. Flow is treated in the FLOW keyword block. The most important inputs concern molecular diffusion and hydrodynamic dispersion, along with transport properties of the medium (e.g., the "cementation factor" or tortuosity) that affect these processes.

Fix_diffusion

Keyword followed by a value for the molecular diffusion coefficient in the space and time units set by the *space_units* and *time_units* keywords within the TRANSPORT block.

Syntax*:* **fix_diffusion** *value*

Default: *0.0 (default units of m^2/yr)*

Explanation: If this parameter is set, then additional instructions to calculate diffusion (*calculate_diffusion*) based on a 25°C value and an activation energy are ignored. If **no** species-specific diffusion coefficients are specified (see *D_25* keyword) With setting of this parameter, all aqueous species are given the same diffusion coefficient and only Fick's law is implemented in the calculation (the electrochemical migration term is zero when all species have the same diffusion coefficient in any case). If any species-specific diffusion coefficients are set, then the full form of the Nernst-Planck equation is solved (see section XXX) and the value given by *fix_diffusion* is used for any species not given a specific diffusion coefficient value with the *D_25* keyword.

Calculate_diffusion

Keyword followed by a value for the molecular diffusion coefficient in the space and time units set by the *space_units* and *time_units* keywords within the TRANSPORT block.

Syntax: **calculate_diffusion** *value*

Default: *0.0 (default units of* m^2*/yr)*

Explanation: This parameter is used to calculate a diffusion coefficient at 25°C. Diffusion coefficients at other temperatures are calculated with the value set in the *diffusion_activation* keyword The fix_diffusion keyword over-rides this keyword (for clarity, both should not normally be set).

Cementation_exponent

Keyword followed by a value for the cementation exponent in Archie's Law.

Syntax: **cementation_exponent** *value*

Default: *1.0*

Explanation: This parameter provides an exponent to be used in Archie's Law to specify a porosity-dependent tortuosity according to:

$$D_{eff} = \phi^m D_w ,$$

where m is the cementation exponent, D_w is the diffusion coefficient in water, and D_{eff} is the effective diffusion coefficient in porous medium. As defined here, m = 1 would be the formulation in porous media in the case where there is no tortuosity effect and the diffusion coefficient is corrected only for the cross-sectional area actually made up of pores. According to this formulation, the formation factor would be defined as:

$$F = \phi^{-m}$$

Formation_factor

Keyword followed by a value for the formation factor.

Syntax: **formation_factor** *value*

Default: *1.0*

Explanation: This parameter provides an a formation factor, which is defined as:

$$D_{eff} = \frac{D_w}{F},$$

where F is the formation factor, D_w is the diffusion coefficient in water, and D_{eff} is the effective diffusion coefficient in porous medium. According to the definition of the formation factor in use in the marine geochemistry literature, for example, the cementation exponent should be set to 0 (i.e, the porosity dependence of the effective diffusion coefficient is incorporated into the formation factor.

Diffusion_activation

Keyword followed by a value for the molecular diffusion activation energy in kcal/mole.

Syntax*:* **diffusion_threshold** *value*

Default: *5.0 (default units of kcal/mole)*

Explanation: This parameter sets the activation energy to be used in calculating diffusion coefficients at temperatures other than 25°C.

D_25

Keyword specifying a diffusion coefficient at 25°C for a specific aqueous species.

Syntax*:* **D_25 species_name** *value*

Default: *value set in fix_diffusion keyword*

Explanation: This keyword provides a diffusion coefficient in the units based on the space_units and time_units keywords set in the TRANSPORT block. When even one species_specific diffusion coefficient is given, the code (GIMRT only) solves the Nernst-Planck equation, which includes an explicit calculation of electrochemical migration. In this case, any species not listed with this keyword are given the diffusion coefficient specified in the *fix_diffusion* keyword.

Gas_diffusion

Keyword followed by a value for the gas diffusion coefficient.

Syntax*:* **gas_diffusion** *value*

Default: *0.0 (default units of m^2/yr)*

Explanation: This keyword sets a value for the gas diffusion coefficient in the units given in the *time_units* and *space_units* keywords set within the TRANSPORT block.

Dispersivity

Keyword followed by a value for the gas diffusion coefficient.

Syntax*:* **dispersivity** <*longitudinal value*> < *transverse value*>

Default: *0.0 (default units of m)*

Explanation: This keyword sets a value for the longitudinal (first entry) and transverse (second value) dispersivities. Space units may be changed with the *space_units* keyword set within the TRANSPORT block. If the problem is one-dimensional, the transverse dispersivity may be omitted.

Constant_tortuosity

Keyword followed by a value for the tortuosity, or by a string giving the name of the tortuosity option.

Syntax*:* constant_tortuosity *value* or constant_tortuosity *option*

Default: *1.0*

Explanation: This keyword sets a constant value for the tortuosity, which is then applied throughout the spatial domain. Definitions of the tortuosity vary—in most cases, it does not include the porosity that multiplies the diffusion coefficient and this is the convention adopted here. The tortuosity is also considered here to be a number less than 1 (some formulations divide by the tortuosity, so that it is always greater than 1). The tortuosity (τ) as used in CrunchFlow, therefore, can be viewed as a parameter which allows an effective diffusion coefficient in porous media (D_{PM}) to be calculated from the diffusion coefficient in pure water (D_w) and the porosity according to:

$$D_{PM} = \phi \tau D_w .$$

If an option rather than a numerical value is given, the code checks the option specified against the list currently implemented. At the time of this manual, the only option that is accepted is *CostaRica*, which is a porosity-tortuosity option to published soon. With an option set, the tortuosity is calculated dynamically at run time (so the keyword *constant_tortuosity* is a bit of a misnomer here), but it avoids a read of tortuosity from an input file (*read_tortuosity*) or by field (*tortuosity*).

Read_TortuosityFile

Keyword followed by a file name containing tortuosity values over the entire spatial domain.

Syntax*:* **read_TortuosityFile** *filename format*

Backwards compatibility: **read_tortuosity**

Default: *None*

Explanation: This keyword provides a filename for a file containing tortuosity values over the entire spatial domain. The tortuosity is considered here to be a number less than 1 (some formulations divide by the tortuosity, so that it is always greater than 1). The tortuosity (τ) as used in CrunchFlow, therefore, can be viewed as a parameter which allows an effective diffusion coefficient in porous media (D_{PM}) to be calculated from the diffusion coefficient in pure water (D_w) and the porosity according to:

$$D_{PM} = \phi \tau D_w .$$

The format of the file depends on the format specified after the file name, but the values of the tortuosity are assumed to be listed with NX varying first, then NY, and then NZ. For those familiar with FORTRAN and using the *SingleColumn* format (the default), the actual source code is:

```
Do jz = 1,nz
  Do jy = 1,ny
    Do jx = 1,nx
```

```
          READ(52,*) tortuosity(jx,jy,jz)
        End Do
      End Do
    End Do
```

Tortuosity

Keyword used to specify tortuosity by zone in the input file.

Syntax*:* **tortuosity** *value zone jxbegin-jxend jybegin-jyend jzbegin-jzend*

or **tortuosity** *value default*

Default: *1.0*

Explanation: The *tortuosity* keyword uses an input format similar to other parameters specified by zone (e.g., pressure, permeability). The keyword is followed by the value of the tortuosity, which is then followed by either the label *zone* or *default*. If *zone* is specified, then the code expects the X, Y, and Z coordinates in the form of a beginning JX, and ending JX, a beginning JY and an ending JY, and a beginning JZ and an ending JZ, in each case separated by a hyphen. If *default* is specified instead of *zone*, the code will initialize the entire spatial domain (i.e., all grid cells) to the value. More than one specification of *tortuosity* can be (and normally is) provided, for example

```
tortuosity  0.1   default
tortuosity  0.01  zone  1-20 1-1 1-1
tortuosity  0.02  zone  1-1 1-20 1-1
```

Zones specified later in the sequence overwrite zones specified earlier—in the example above, the default specification sets the tortuosity value to 0.1 initially in all of the grid cells, while the next tortuosity keyword changes the value to 0.01 in grid cells JX=1 to JX=20. *The convention adopted here is that all three dimensions must be specified, even in a one-dimensional problem.*

The tortuosity is considered here to be a number less than 1 (some formulations divide by the tortuosity, so that it is always greater than 1). The tortuosity (τ) as used in CrunchFlow, therefore, can be viewed as a parameter which allows an effective diffusion coefficient in porous media (D_{PM}) to be calculated from the diffusion coefficient in pure water (D_w) and the porosity according to:

$$D_{PM} = \phi(jx, jy, jz)\tau(jx, jy, jz)D_w .$$

Anisotropy_ratioY

Keyword followed by a value defining the anisotropy ratio in the Y direction relative to the value in the X direction.

Syntax*:* **Anisotropy_ratioY** *value*

Default: *1.0*

Explanation: This keyword defines the ratio of the diffusion coefficient in the Y direction relative to the diffusion coefficient in the X direction. The anisotropy term, therefore, is dimensionless. If the anisotropy factor in the Y direction is defined as ε_Y, then the effective diffusion coefficient in the Y direction (JY) is given by:

$$D_{PM} = \varepsilon_Y \phi \tau D_w .$$

Anisotropy_ratioZ

Keyword followed by a value defining the anisotropy ratio in the Z direction relative to the value in the X direction.

Syntax*:* **anisotropy_ratioZ** *value*

Default: *1.0*

Explanation: This keyword defines the ratio of the diffusion coefficient in the Z direction relative to the diffusion coefficient in the X direction. The anisotropy term, therefore, is dimensionless. If the anisotropy factor in the Z direction is defined as ε_Z, then the effective diffusion coefficient in the Z direction (JZ) is given by:

$$D_{PM} = \varepsilon_Z \phi \tau D_w .$$

Threshold_porosity

Keyword followed by a value defining a porosity threshold separating two different tortuosity values.

Syntax*:* **threshold_porosity** *value*

Default: *0.0*

Explanation: This keyword sets a threshold porosity, either side of which the tortuosity takes on differing values given by the keywords *tortuosity_below* and *tortuosity_above*.

Tortuosity_below

Keyword followed by a value giving the tortuosity below the porosity threshold set in the keyword *threshold_porosity*..

Syntax*:* **tortuosity_below** *value*

Default: *0.0 (only used where threshold_porosity is set)*

Explanation: This keyword sets the value of the tortuosity to be used below the threshold porosity. This keyword is only used if *threshold_porosity* is set.

Tortuosity_above

Keyword followed by a value giving the tortuosity above the porosity threshold set in the keyword *threshold_porosity*..

Syntax*:* **tortuosity_above** *value*

Default: *0.0 (only used where threshold_porosity is set)*

Explanation: This keyword sets the value of the tortuosity to be used above the threshold porosity. This keyword is only used if *threshold_porosity* is set.

FLOW

Constant_flow

Keyword followed by a value(s) giving the X, Y, and Z components of the Darcy flux.

Syntax*:* **constant_flow** *qx <qy> <qz>*

Default: *0.0 ($m^3/m^2/yr$)*

Explanation: This keyword is used to set a constant Darcy flux. As a vector, it includes X, Y, and Z components. In a one-dimensional problem in X, the Y and Z velocities are optional. Units are changed inside the FLOW block with the *space_units* and *time_units* keywords.

Constant_gasflow

Keyword followed by a value(s) giving the X, Y, and Z components of the gas flux.

Syntax*:* **constant_gasflow** *qxgas <qygas> <qzgas>*

Default: *0.0 ($m^3/m^2/yr$)*

Explanation: This keyword is used to set a constant gas flux. As a vector, it includes X, Y, and Z components. In a one-dimensional problem in X, the Y and Z velocities are optional. Units are changed inside the FLOW block with the *space_units* and *time_units* keywords.

Pump

Keyword followed by a value(s) giving pumping rate, the name of the geochemical condition to be used in the case of an injection well, and its grid coordinates.

Syntax*:* **pump rate label** *jx <jy> <jz>*

Default: *None*

Explanation: This keyword beginning with *pump* sets a pumping rate (positive for injection, negative for withdrawal) in units of l/sec and assigns a geochemical condition to it. This is followed by the coordinates of the well, with JY and JZ being optional in the case of a one-dimensional problem. The geochemical condition is identified by its label, although the condition is only used in the case of an injection well.

Gaspump

Keyword followed by a value(s) giving gas pumping rate, the name of the geochemical condition to be used in the case of an injection well, and its grid coordinates.

Syntax*:* **gaspump rate label** *jx <jy> <jz>*

Default: *None*

Explanation: This keyword beginning with *gaspump* sets a gas pumping rate (positive for injection, negative for withdrawal) in units of l/sec and assigns a geochemical condition to it. This is followed by the coordinates of the well, with JY and JZ being optional in the case of a one-dimensional problem. The geochemical condition is identified by its label, although the condition is only used in the case of an injection well.

Infiltration

Keyword followed by a value(s) giving the infiltration or recharge rate, the name of the geochemical condition to be used, and the coordinate direction (X, Y, or Z).

Syntax*:* **infiltration rate label** *coordinate*

Default: *None*

Explanation: This keyword specifies an infiltration rate which is distributed over the boundary of the system (at JX, JY, or JZ = 0). The geochemical condition to be used is given by its **label**.

Read_VelocityFile

Keyword followed by a name for a file containing velocity vectors over the entire spatial domain.

Syntax*:* **read_VelocityFile** *filename format*

Backwards compatible*:* **read_velocity**

Default: *None*

Explanation: This keyword provides a filename prefix for ASCII files containing velocity vector values over the entire spatial domain. The extensions *vx*, *vy*, and *vz* are assumed to indicate the files containing the X, Y, and Z velocities, respectively. For example, the syntax

```
read_VelocityFile  copperleach   SingleColumn
```

would open and attempt to read the files:

```
copperleach.vx
copperleach.vy
copperleach.vz
```

using the SingleColumn format (see Section 7.1.2). Units are set with the *space_units* and *time_units* keywords—otherwise, default units are m^3/m^2/yr.

The format of the file depends on the format chosen. In the absence of a specification of the file format, a single column (*SingleColumn*) format is assumed. Velocities are defined at grid interfaces (not at the center of the nodes), so NX+1 X velocities are needed in the X coordinate direction, NY+1 Y velocities in the Y coordinate direction, and NZ+1 Z velocities in the Z coordinate direction. CrunchFlow uses the convention that velocities for a particular grid cell are defined at the right hand interface (the so-called "right hand rule"). So, for example, the velocity, *qx(2,1,1)* is assumed to be the X velocity at the interface between grid cell 2 and 3 (Figure XX). Accordingly, the first entry in the *velocityfile.vx* file will be the X velocity defined at the interface between grid cell (0,1,1) and (1,1,1). So, in the case of where *ContinuousRead* is specified as the file format, the code would read:

```
READ(52,*) (qx(jx,jy,jz),jx=0,nx),jy=1,ny),jz=1,nz)
READ(53,*) (qy(jx,jy,jz),jx=1,nx),jy=0,ny),jz=1,nz)
READ(54,*) (qz(jx,jy,jz),jx=1,nx),jy=1,ny),jz=0,nz)
```

Read_GasVelocityFile

Keyword followed by a name for a file containing gas velocities over the entire spatial domain.

Syntax*:* **read_GasVelocityFile** *filename format*

Backwards compatible*:* **read_gasvelocity**

Default: *None*

Explanation: This keyword provides a filename prefix for ASCII files containing gas velocities over the entire spatial domain. The extensions *vx*, *vy*, and *vz* are assumed to indicate the files containing the X, Y, and Z velocities, respectively. For example, the syntax

```
read_GasVelocityFile  co2stream  ContinuousRead
```

would open and attempt to read the files:

```
co2stream.vx
co2stream.vy
co2stream.vz
```

using the ContinuousRead format (see Section 7.1.2) Units are set with the space_units and time_units keywords—otherwise, default units are $m^3/m^2/yr$.

The format of the file depends on the format chosen. In the absence of a specification of the file format, a single column (*SingleColumn*) format is assumed. Velocities are defined at grid interfaces (not at the center of the nodes), so NX+1 X velocities are needed in the X coordinate direction, NY+1 Y velocities in the Y coordinate direction, and NZ+1 Z velocities in the Z coordinate direction. CrunchFlow uses the convention that velocities for a particular grid cell are defined at the right hand interface (the so-called "right hand rule"). So, for example, the gas velocity, *qxgas(2,1,1)* is assumed to be the X velocity at the interface between grid cell 2 and 3 (Figure XX). Accordingly, the first entry in the gas*velocityfile.vx* file will be the X velocity defined at the interface between grid cell (0,1,1) and (1,1,1). So, in the case of where *ContinuousRead* is specified as the file format, the code would read:

```
READ(52,*) (qxgas(jx,jy,jz),jx=0,nx),jy=1,ny),jz=1,nz)
READ(52,*) (qygas(jx,jy,jz),jx=1,nx),jy=0,ny),jz=1,nz)
READ(52,*) (qzgas(jx,jy,jz),jx=1,nx),jy=1,ny),jz=0,nz)
```

Calculate_flow

Keyword followed by *true* or *false,* which indicates whether flow will be calculated within the code or not.

Syntax*:* **calculate_flow** *logical*

Default: *False*

Explanation: This keyword indicates whether flow will be calculated according to Darcy's Law or not. At this time, only fully saturated flow calculations are possible. Once this logical is set to *true*, the code will look for inputs on the pressure and permeability fields.

Read_PermeabilityFile

Keyword followed by a file name containing permeability values over the entire spatial domain.

Syntax: **read_PermeabilityFile** *filename format*

Backwards compatible: **read_permeability**

Default: *None*

Explanation: This keyword provides a filename prefix for ASCII files containing permeability values over the entire spatial domain. The extensions *x*, *y*, and *z* are assumed to indicate the files containing the X, Y, and Z permeabilities, respectively. For example, the syntax

```
read_PermeabilityFile  copperleach  SingleColumn
```

would open and attempt to read the files:

```
copperleach.x
copperleach.y
copperleach.z
```

using the *SingleColumn* (see Section 7.1.2) format. Units of the permeability are m^2.

The format of the file depends on the file format specified. In none is provided, a single column format consisting of a single permeability per line, with NX varying first, then NY, and then NZ is assumed. Unlike the fluid velocity, permeabilities are defined at the center of the nodes, since at least in some cases they are linked to properties like the porosity, which in turn may be affected by mineral dissolution and precipitation. Interface permeabilities are calculated as harmonic means between adjacent cells. For example, between nodes *i* and *i+1*, the harmonic mean of the permeabililty is given by:

$$k_{i+1/2} = \frac{2\left(k_i k_{i+1}\right)}{\left(k_i + k_{i+1}\right)}.$$

Since permeabilities need to be defined at the boundaries to determine the amount of flow across them, “ghost cell” X permeabilities are required at the 0 and nx+1 nodes, and so on. For 1D problems, *permy* and *permz* files are not needed.

For those familiar with FORTRAN, the actual source code in the case where a *SingleColumn* format is specified is given by (compare to the *ContinuousRead* format in the case of the *read_VelocityFile* keyword):

```
DO jz = 1,nz
  DO jy = 1,ny
    DO jx = 0,nx+1
      READ(52,*) permx(jx,jy,jz)
    END DO
  END DO
END DO
```

```
DO jz = 1,nz
  DO jy = 0,ny+1
    DO jx = 1,nx
      READ(53,*) permy(jx,jy,jz)
    END DO
  END DO
END DO

DO jz = 0,nz+1
  DO jy = 1,ny
    DO jx = 1,nx
      READ(54,*) permz(jx,jy,jz)
    END DO
  END DO
END DO
```

Permeability_x

Keyword used to specify the permeability in the X coordinate direction by zone in the input file.

Syntax: **permeability_x** *value zone jxbegin-jxend jybegin-jyend jzbegin-jzend*

or **permeability_x** *value default*

Default: *None*

Explanation: The *permeability_x* keyword uses an input format similar to other parameters specified by zone (e.g., pressure, tortuosity). The keyword is followed by the value of the X permeability, which is then followed by either the label *zone* or *default*. If *zone* is specified, then the code expects the X, Y, and Z coordinates in the form of a beginning JX, and ending JX, a beginning JY and an ending JY, and a beginning JZ and an ending JZ, in each case separated by a hyphen. If *default* is specified instead of *zone*, the code will initialize the entire spatial domain (i.e., all grid cells) to the value. More than one specification of *permeability_x* can be (and normally is) provided. For example, in a 2D system with 20 grid cells in the X coordinate direction and 20 in the Y coordinate direction, a higher X permeability zone running down the center of the domain could be specified with

```
permeability_x    1.E-13  default
permeability_x    1.E-12  zone  0-21 10-10 1-1
```

Note that values of the X permeability at grid cell 0 and NX+1 have been specified, since the harmonic mean of the permeability is calculated at the system boundaries using these “ghost cell” values. Zones specified later in the sequence overwrite zones specified earlier—in the example above, the default specification sets the *permeability_x* value to 1.E-13 m^2 initially in all of the grid cells, while the next *permeability_x* keyword creates a higher permeability zone in grid cells JY=10 over the entire X length of the domain. To create a no-flow boundary at JX=0 over the entire Y length, one could use:

```
permeability_x    1.E-13  default
permeability_x    1.E-12  zone  0-0 1-20 1-1
```

or more selectively with something like:

```
permeability_x     1.E-13  default
permeability_x     1.E-12  zone  0-0 1-8 1-1
permeability_x     1.E-12  zone  0-0 12-20 1-1
```

which would then create a no-flow boundary at JX=0 except for JY nodes 9, 10, and 11.

The convention adopted here is that all three dimensions must be specified as zones, even in a one-dimensional problem, although permeability_y and permeability_z are not needed.

Permeability_y

Keyword used to specify the permeability in the Y coordinate direction by zone in the input file.

Syntax*:* **permeability_y** *value zone jxbegin-jxend jybegin-jyend jzbegin-jzend*

or **permeability_y** *value default*

Default: *None*

Explanation: The *permeability_y* keyword uses an input format similar to other parameters specified by zone (e.g., pressure, tortuosity). See the keyword *permeability_x* for details on how this keyword is used to specify permeabilities by zone. *The convention adopted here is that all three dimensions must be specified, even in a one-dimensional problem.*

Permeability_z

Keyword used to specify the permeability in the Z coordinate direction by zone in the input file.

Syntax*:* **permeability_z** *value zone jxbegin-jxend jybegin-jyend jzbegin-jzend*

or **permeability_z** *value default*

Default: *None*

Explanation: The *permeability_z* keyword uses an input format similar to other parameters specified by zone (e.g., pressure, tortuosity). See the keyword *permeability_x* for details on how this keyword is used to specify permeabilities by zone. *The convention adopted here is that all three dimensions must be specified, even in a one-dimensional problem.*

Gravity

Keyword used to indicate the coordinate direction of the gravity vector.

Syntax*:* **gravity** *<Angle to X > <Angle to Y > <Angle to Z > direction*

Default: *90°*

Explanation: This keyword provides the angle relative to the X, Y, and Z coordinate directions for the gravity vector. The influence of gravity is then calculated from

$$g = \cos\theta,$$

where θ is the angle of the gravity vector relative to the specific coordinate direction (X, Y, or Z). Direction can be either *down* or *up* and is used to indicate whether the direction the gravity vector is assumed to operate (for example, toward X[NX] or toward X[0]). So, for example, the following input

```
        gravity    90.0  0.0  90.0  down
```

would be used to set the gravity vector in the Y direction, with gravity oriented down (toward JY = 0).

Pressure

Keyword used to specify the fluid pressure by zone in the input file.

Syntax*:* **pressure** *value zone jxbegin-jxend jybegin-jyend jzbegin-jzend [fix]*

or **pressure** *value default*

Default: *None*

Explanation: The *pressure* keyword uses an input format similar to other parameters specified by zone (e.g., tortuosity, permeability). The keyword is followed by the value of the fluid pressure in units of Pascals, which is then followed by either the label *zone* or *default*. If *zone* is specified, then the code expects the X, Y, and Z coordinates in the form of a beginning JX, and ending JX, a beginning JY and an ending JY, and a beginning JZ and an ending JZ, in each case separated by a hyphen. If *default* is specified instead of *zone*, the code will initialize the entire spatial domain (i.e., all grid cells) to the value. More than one specification of *pressure* can be (and normally is) provided, for example

```
        pressure    1000.0  default
        pressure    0.0     zone  1-20 21-21 1-1 fix
```

Zones specified later in the sequence overwrite zones specified earlier—in the example above, the default specification sets the fluid pressure to 1000 Pa initially in all of the grid cells, while the next keyword sets the pressure at JY = 21 to zero. In the case of a 20 by 20 X-Y simulation, the grid cell 21 corresponds to node NY+1 and is therefore a ghost cell. This is the normal way in which fixed pressure boundary conditions are set. Following the zone coordinates with *fix* instructs the code to lock the value in for the course of the simulation—not including *fix* would mean that the pressure is only set as an initial condition, after which it is free to evolve according to the physics of the problem. *The convention adopted here is that all three dimensions must be specified, even in a one-dimensional problem.*

To create a hydrostatic pressure gradient in the Y direction, with depth increasing with increasing JY, one could use:

```
        pressure    1000.0  default
        pressure    0.0     zone  1-20 0-0 1-1 fix
        gravity     90.0  0.0  90.0  up
```

With this input, the code would time step until the hydrostatic pressure gradient was achieved in the absence of other forcings. To initialize the system to hydrostatic pressure before time stepping begins, use:

```
        pressure    1000.0  default
        pressure    0.0     zone  1-20 0-0 1-1 fix
```

```
gravity     90.0  0.0  90.0  up
initialize_hydrostatic  true
```

This will create initially hydrostatic conditions even if pressure boundary conditions or source/sink terms are such that hydrostatic conditions will not prevail once time stepping begins.

Initialize_hydrostatic

Keyword followed by *true* or *false* which selects whether or not to initialize the domain to hydrostatic conditions.

Syntax*:* **initialize_hydrostatic** *logical*

logical is a standard Fortran logical (true or false).

Default: *False*

Explanation: The keyword parameter *initialize_hydrostatic* is followed by *true* or *false* (or *yes* or *no*) and is used to select, when true, the option to initialize the pressure field to hydrostatic conditions before time stepping begins. When set to true, hydrostatic conditions will be initialized no matter what other pressure boundary conditions or source/sink terms take effect once time stepping begins. However, a coordinate direction for the gravity vector must be selected with the *gravity* keyword and appropriate boundary conditions (typically just a fixed pressure at one end of the domain).

TEMPERATURE

This keyword block is used to set various temperature parameters. A full calculation of the temperature will be added soon, but in the meantime, temperature can be set block by block using individual geochemical conditions, or by reading temperature values from a file.

Set_temperature

Keyword followed by a value giving the default temperature.

Syntax*:* **set_temperature** *value*

Default: 25ºC

Explanation: The keyword parameter *set_temperature* is used to set a global value for the temperature. The value provided here can be overwritten by specifications in individual geochemical conditions, provided these conditions are distributed in space as initial conditions.

Temperature_gradient

Keyword followed by a value giving the temperature gradient.

Syntax*:* **temperature_gradient** *value*

Default: 0.0

Explanation: The keyword parameter *temperature_gradient* can be used to set a temperature gradient in units of ºC/m. A positive value means that the temperature gradient will increase from grid point 1 to grid point NX. The temperature gradient, when input in this way, operates only in the X direction.

Read_TemperatureFile

Option to specify a file with temperatures to be read.

Syntax*:* **read_temperaturefile** *filename format*

filename gives the name of the file(up to 132 characters) contining the liquid saturation distributed over the entire spatial domain.

Default: None.

Explanation: This provides the name of a file containing temperatures distributed in space to be read. This option supersedes all other temperature specifications. The format of the file depends on the optional format specified (see Section 7.1.2). If no format is specified, the code assumes a single column (SingleColumn) of values in the order 1-NX, 1-NY, 1-NZ:

```
DO jz = 1,nz
  DO jy = 1,ny
    DO jx = 1,nx
      READ(52,*) t(jx,jy,jz)
    END DO
  END DO
END DO
```

POROSITY

This keyword block is used to set parameters and options related to the treatment of the porosity. The main contrast is between simulations in which the porosity is fixed and those in which it evolves as the mineral volume fractions evolve.

Fix_porosity

Keyword followed by a value giving the default fixed porosity.

Syntax*:* **fix_porosity** *value*

Default: If not provided, the code calculates porosity based on the mineral volume fractions.

Explanation: The keyword parameter *fix_porosity* is used to set a global fixed porosity for a simulation. When provided, it disables the calculation of porosity based on the sum of the volume fractions. Accordingly, no porosity update based on these quantities is possible. When this parameter is set in the POROSITY keyword block, *set_porosity* keywords within individual geochemical conditions can be used once the geochemical condition is distributed in space within the INITIAL_CONDITION keyword block.

Minimum_porosity

Keyword giving the minimum porosity value that is allowed as the mineral volume fractions evolve.

Syntax*:* **minimum_porosity** *value*

Default: 10^{-14}

Explanation: This keyword is only used to set a minimum to which the porosity can evolve as the mineral volume fractions change. This option is only used where *porosity_update* is true and neither *fix_porosity* nor *read_porosity* are set.

Porosity_update

Keyword followed by a logical (true or false) used to determine whether the porosity is updated as the mineral volume fractions evolve.

Syntax*:* **porosity_update** *logical*

Default: *False*

Explanation: This keyword is only used where neither the fix_porosity or read_porosity keywords have been set. When true, it instructs the code to update the porosity as the mineral volume fractions evolve according to

$$\phi = \sum_{m=1}^{Nm} 1 - \phi_m,$$

where N_m is the number of minerals in the system and ϕ_m is the volume fraction of the individual mineral.

Read_PorosityFile

Keyword followed by a file name containing permeability values over the entire spatial domain.

Syntax*:* **read_ PorosityFile** *filename format*

Backwards compatible: **read_ porosity**

Default: *None*

Explanation: This keyword provides the name of file containing porosity values defined at the center of the grid cell for the entire spatial domain. The format of the file depends on the file format specified. In none is provided, a single column format consisting of a single porosity per line, with NX varying first, then NY, and then NZ is assumed.

PEST

CreatePestInstructionFile

Keyword followed by *true* or *false* which selects whether or not to create a PEST instruction (*PestExchange.ins*) file.

Syntax*:* **CreatePestInstructionFile** *logical*

Default: *False*

Explanation: This logical determines whether a *PestExchange.ins* file is created with instructions for PEST to read output on cation exchange from CrunchFlow. The format specified within the instruction file will match the format used by CrunchFlow, with the cation exchange species listed in the order given the *exchange* keyword within the PEST keyword block. Once created, this option should be turned off so as to avoid writing over the *PestExchange.ins* file.

CreatePestExchangeFile

Keyword followed by a file name.

Syntax: **CreatePestInstructionFile** *filename*

Default: *PestExchange.out*

Explanation: This keyword is used to set the filename to be used for exchange output used in PEST runs. This option is useful where multiple input files are being run together, thus requiring different filenames for the output.

Exchange

Keyword followed by concentration units followed by a list of exchange species to be written out to a file PestExchange.out.

Syntax: **Exchange** *units 'exchange species list'*

Default: *None*

Explanation: This keyword indicates the list of species participating in exchange reactions (e.g., Na^+, Ca^{2+}, Mg^{2+}, …) to be written out to the file *PestExchange.out*, which can then be used as "data" or "observations" in a PEST run using CrunchFlow. If the keyword *CreatePestInstructionFile* is also set as true, then the format for the exchange species output will be written to a PEST instruction file: *PestExchange.ins*. The exchange species list is preceded by a specification of the units, which may include:

Units	**Alternate Designation**
mol/g	mole/g, moles/g
mmol/g	mmole/g, mmoles/g
umol/g	umole/g, umoles/g
equiv/g	eq/g, equivalents/g
mequiv/g	milliequiv/g, meq/g, milliequivalents/g
uequiv/g	microequiv/g, ueq/g, microequivalents/g

EROSION

Read_BurialFile

Keyword followed by a file name containing erosion/burial values over the X direction.

Syntax: **read_BurialFile** *filename format*

Backwards compatibility: **read_burial**

Default: *None*

Explanation: This keyword provides a name for a file containing burial/erosion rates values in the X coordinate direction. Other coordinates (Y and Z) are not currently supported. Units are determined by *space_units* and *time_units* specifications within the EROSION keyword block. Default units are m/yr.

The format of the file depends on the format specified after the file name, but the values of the erosion/burial rates are assumed to be listed with NX varying first, then NY, and then NZ. For those familiar with FORTRAN and using the *SingleColumn* format (the default), the actual source code is:

```
Do jx = 0,nx
  READ(52,*) FluidBuryX(jx,jy,jz),SolidBuryX(jx,jy,jz)
End Do
```

Acknowledgements

Many people have delayed the completion of the manual. When I think of anybody who has contributed to speeding its completion, I will list them here.

Literature Cited

Aagaard P. and Helgeson H.C. (1982) Thermodynamic and kinetic constraints on reaction rates among minerals and aqueous solutions, I, Theoretical considerations. *Amer. J. Sci.* , 237-285.

Aris R. (1965) Prolegomena to the rational analysis of systems of chemical reactions. *Arch. Rational Mech. and Anal.* , 81-99.

Bear J. (1972) *Dynamics of Fluids in Porous Media*. American Elsevier, New York, 764 p.

Bear J. (1979) *Hydraulics of Groundwater*. McGraw–Hill, New York, 569 p.

Berner R.A. (1980) *Early Diagenesis: A Theoretical Approach*. Princeton Univ. Press, Princeton, 241p.

Bowen R.M. (1968) On the stoichiometry of chemically reacting materials. *Arch. Rational Mech. Anal.* , 114-124.

Bruges E.A., Latto B., and Ray A.K. (1966) New correlations and tables of the coefficient of viscosity of water and steam up to 1000 bar and 1000 $^\circ$C. *Internat. Jour. Heat and Mass Transfer* , 465-480.

Daus A.D. and Frind E.O. (1985) An alternating direction Galerkin technique for simulation of contaminant transport in complex groundwater systems *Water Resources Res.* , 653-664.

Dullien F.A.L. (1979) *Porous Media*. Academic Press, San Diego, 396 p.

Frind E.O. and Germain D. (1986) Simulation of contaminant plumes with large dispersive contrast: Evaluation of alternating direction Galerkin models. *Water Resources Res.* , , 1857-1873.

Gupta, A.D., Lake, L.W., Pope, G.A., Sephernoori, K., and King, M.J. (1991). High–resolution monotonic schemes for reservoir fluid flow simulation. *In Situ.*, , 289-317.

Helgeson H.C. (1968) Evaluation of irreversible reactions in geochemical processes involving minerals and aqueous solutions-I. Thermodynamic relations. *Geochim. Cosmochim. Acta* , 853-877.

Helgeson H.C. (1979) Mass transfer among minerals and hydrothermal solutions. In *Geochemistry of Hydrothermal Ore Deposits* (ed. H.L. Barnes), John Wiley and Sons, New York, 568-610.

Helgeson H.C. and Kirkham D.H. (1974) Theoretical predictions of the thermodynamic behavior of aqueous electrolytes at high pressures and temperatures; I: Summary of the thermodynamic/electrostatic properties of the solvent. *Amer. J. Sci.* , 1089-1198.

Helgeson H.C., Garrels R.M., and MacKenzie F.T. (1969) Evaluation of irreversible reactions in geochemical processes involving minerals and aqueous solutions-II. Applications. *Geochim. Cosmochim. Acta* , 455-482.

Hindmarsh, A.C. (1977) Solution of block–tridiagonal systems of linear algebraic equations. UCID-30150.

Hooyman G.J. (1961) On thermodynamic coupling of chemical reactions. *Proc. Nat. Acad. Sci.* , 1169-1173.

Johnson J. W., Oelkers E. H. and Helgeson H. C. (1992) *SUPCRT92:* A software package for calculating the standard molal thermodynamic properties of minerals, gases, aqueous species, and reactions from 1 to 5000 bars and 0 $^\circ$ to 1000 $^\circ$ C, *Computers in Geosciences*, in press.

Kirkner D.J. and Reeves H. (1988) Multicomponent mass transport with homogeneous and heterogeneous chemical reactions: Effect of chemistry on the choice of numerical algorithm. I. Theory. *Water Resources Res.* , 1719-1729.

Lapidus L. and Pinder G.F. (1982) *Numerical Solution of Partial Differential Equations in Science and Engineering*, John Wiley and Sons, New York, 677 p.

Lasaga A.C. (1981) Rate laws in chemical reactions. In *Kinetics of Geochemical Processes* (ed. A.C. Lasaga and R.J. Kirkpatrick), Rev. Mineral. , 135-169.

Lasaga A.C. (1984) Chemical kinetics of water-rock interactions. *J. Geophys. Res.* , 4009-4025.

Lichtner P.C. (1985) Continuum model for simultaneous chemical reactions and mass transport in hydrothermal systems. *Geochim. Cosmochim. Acta* , 779-800.

Lichtner P.C. (1988) The quasi-stationary state approximation to coupled mass transport and fluid-rock interaction in a porous medium. *Geochim. Cosmochim. Acta* , 143-165.

Lichtner P.C. (1992) Time–space continuum description of fluid/rock interaction in permeable media. *J. Geophys. Res.* , 3135-3155.

Marsily G.de (1986) *Quantitative Hydrogeology*. Acad. Press, New York, 440 p.

Patankar S.V. (1980) *Numerical Heat Transfer and Fluid Flow*. Hemisphere Publ., New York, 197 p.

Patel M.K., Cross M., and Markatos N.C. (1988) An assessment of flow oriented schemes for reducing ’false diffusion’. *Int. Journ. Numerical Methods in Engineering* , p. 2279-2304.

Press W.H., Flannery B.P., Teukolsky S.A., and Vetterling W.T. (1986) *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, 818 p.

Reed M.H. (1982) Calculation of multicomponent chemical equilibria and reaction processes in systems involving minerals, gases, and an aqueous phase. *Geochim. Cosmochim. Acta* , 513-528.

Steefel C.I. (1992) Coupled fluid flow and chemical reaction: Model development and application to water–rock interaction. Ph.D. thesis, New Haven, Connecticut, Yale University, 234 p.

Steefel C.I. and Lasaga A.C. (1990) Evolution of dissolution patterns: Permeability change due to coupled flow and reaction. In *Chemical Modeling in Aqueous Systems II* (ed. D.C. Melchior and R.L. Bassett), ACS Symp. Ser. No. 416, 212-225.

Steefel C.I. and Lasaga A.C. (1992) Putting transport into water–rock interaction models. *Geology* , 680-684.

Steefel C.I. and Lasaga A.C. (1994) A coupled model for transport of multiple chemical species and kinetic precipitation/dissolution reactions with application to reactive flow in single phase hydrothermal systems. *American Journal of Science*, , 529-592.

Steefel C.I. and Lichtner P.C. (1994) Diffusion and reaction in rock matrix bordering a hyperalkaline fluid-filled fracture. *Geochim. Cosmochim. Acta* , 3595-3612.

Steefel C.I. and Van Cappellen P. (1990) A new kinetic approach to modeling water–rock interaction: The role of nucleation, precursors, and Ostwald ripening. *Geochim. Cosmochim. Acta* , 2657–2677.

Strang, G.. (1968) On the construction and comparison of difference schemes, *SIAM J. Numer. Anal.* , 506-517.

Valocchi, Albert J. and Malmstead, Michael (1992). Accuracy of operator splitting for advection-dispersion-reaction problems. *Water Resources Res.* bf 28, 1471.

VanderKwaak J.E., Forsyth P.A., MacQuarrie K.T.B., and Sudicky E.A. (1995) WATSOLV Sparse Matrix Iterative Solver Package Versions 1.01. Unpublished report, Waterloo Centre for Groundwater Research, University of Waterloo, Waterloo, Ontario, Canada. 23 p.

Van Zeggeren F. and Storey S.H. (1970) *The Computation of Chemical Equilibria*. Cambridge University Press, Cambridge, 176 p.

Walter A.L., Frind E.O., Blowes D.W., Ptacek C.J., and Molson J.W. (1994). Modeling of multicomponent reactive transport in groundwater: 1. Model development and evaluation. *Water Resources Res.* , 3137-3148.

Wolery T.J., Jackson K.J., Bourcier W.L., Bruton C.J., Viani B.E., Knauss K.G., and Delany J.M. (1990) Current status of the EQ3/6 software package for geochemical modeling. In *Chemical Modeling in Aqueous Systems II* (ed. D.C. Melchior and R.L. Bassett), ACS Symp. Ser. No. 416, 104-116.

Zysset, A., F. Stauffer, and T. Dracos. (1994). Modeling of chemically reactive groundwater transport, *Water Resources Res.* , 2217-2228.