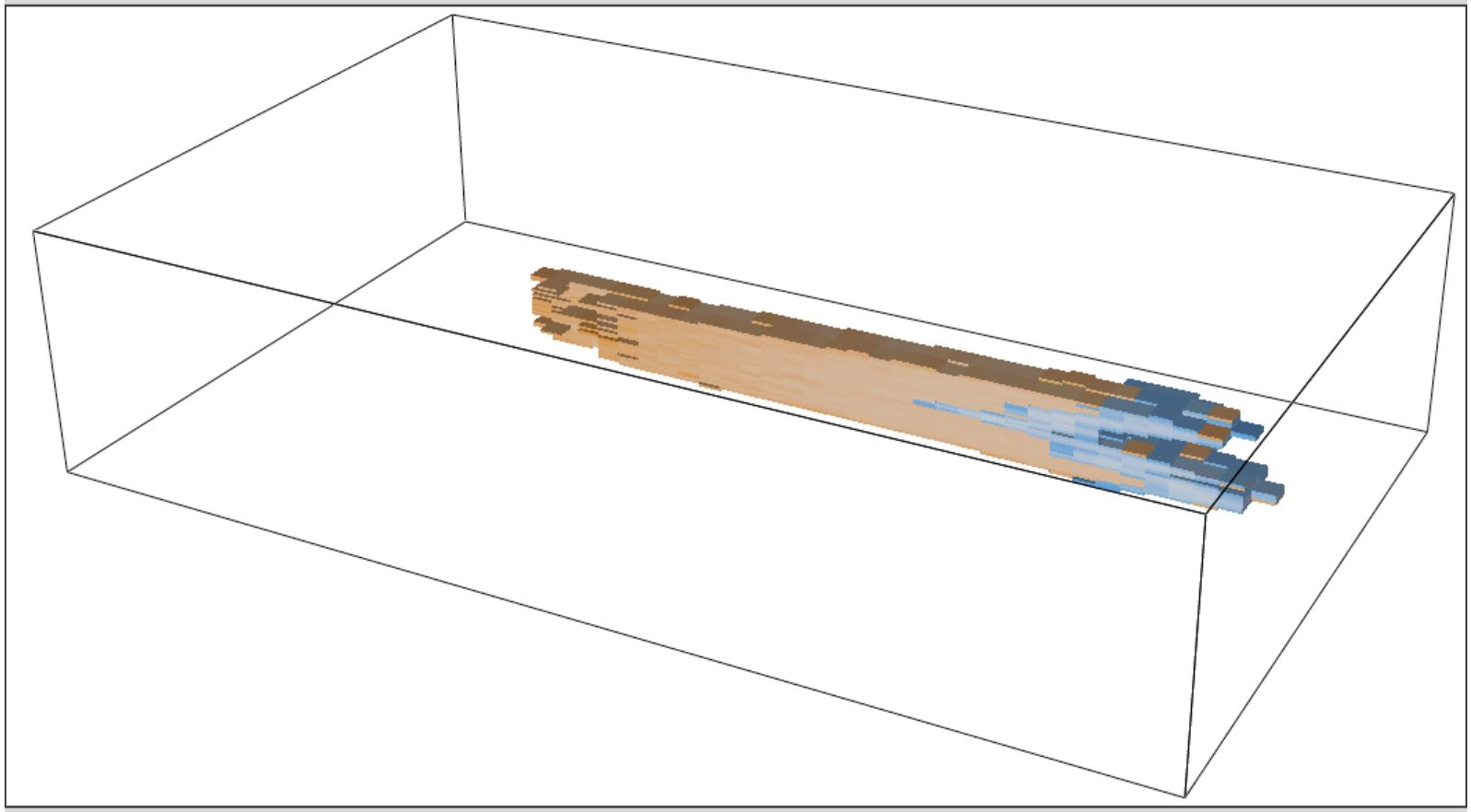
GWMI 2010-01

International Ground Water Modeling Center

SLIM Fast – A numerically-based computer model designed to simulate the migration of dissolved, neutrally buoyant, reactive chemical species in saturated and unsaturated groundwater systems

Constructed using FORTRAN 90/95

March 2010



SLIM-FAST: A User's Manual

Version 4.0
March 2010

Reed M. Maxwell

[1]International Groundwater Modeling Center
Hydrologic Science and Engineering Program
Department of Geology and Geologic Engineering
Colorado School of Mines
1500 Illinois St
Golden, CO 80403
rmaxwell@mines.edu

This manual may be cited as:
Maxwell, R.M. (2010) SLIM-FAST: A User's Manual V.4. GWMI 2010-01, 49 p.

SLIM-FAST: A User's Manual
Version 4.0

1. Introduction

SLIM-FAST is a numerical-based computer model designed to simulate the migration of dissolved, neutrally buoyant, reactive chemical species in saturated groundwater systems. It is based upon an explicit Lagrangian Random Walk Particle Method (RWPM) in which moving clouds of numerical "particles" are used to represent the concentration and spatial distribution of chemical mass as a function of time. In this sense, the particles are a mathematical construct used to represent the material coordinates of the chemical species. They can also be used to describe other distributed characteristics associated with specific material parcels of the mass.

In the RWPM approach, particles are assigned attributes such as position, mass, velocity, diffusivity, chemical species index, or age, which are updated or modified as a function of time. For example, SLIM-FAST uses an externally supplied groundwater velocity field and specified chemical diffusivities as a means to move and disperse (i.e., change the position of) particles over time. Kinetic chemical reactions, such as radioactive decay, may be treated by adjusting the mass, number, or species index of the particles over time. The RWPM approach and related particle-based transport models have been widely applied to solve mass, momentum, and energy transport problems in many disciplines. The mass balance equations and details of the particle model algorithm within SLIM-FAST are given in Appendices A and B.

SLIM-FAST was originally developed as an improved version of the SLIM model (Tompson et al., 1987) that was used in the groundwater transport analyses of Tompson and Gelhar (1990), Tompson and Dougherty (1992), and Tompson et al. (1996, 1998). The current version of SLIM-FAST represents a more current and updated version of the former SLIM and has been used in additional applications described by Maxwell and Kastenberg (1999), Maxwell, et al. (1999), Rubin, et al. (1999), Tompson, et al. (1999, 2005), Pawloski, et al. (2001), Maxwell, et al. (2003), Carle, et al. (2003), and Zavarin, et al. (2004).

1.1 Current Version Capabilities

The current version of SLIM-FAST (v1.0) is based upon the following assumptions, conceptualizations, and specifications:

- The model domain is assumed to be orthogonal and rectilinear with overall dimensions (Lx, Ly, Lz),
- The model domain is assumed to be discretized into a rectilinear grid of (nx, ny, nz) blocks or cells of constant or fixed dimensions (dx, dy, dz). Thus, Lx = nx*dx, Ly = ny*dy, and Lz = nz*dz.
- The model grid is used to represent (i) externally supplied velocity fields, (ii) other externally supplied hydrogeologic or geochemical parameters such as porosity or chemical sorption distribution coefficient, and (iii) gridded concentration fields calculated from particle distributions.
- The underlying transport algorithm is based upon fully-saturated or variably-saturated groundwater flow.

- The transport model can address pure matrix flow, pure fracture flow, or fracture flow with matrix diffusion.
- Groundwater velocity fields may be specified as either steady-state or transient.
- Groundwater velocity fields may be defined from (i) imported Darcy flux components, calculated separately and specified normally across the face of each grid cell and cell-centered porosity values or (ii) imported hydraulic head fields, calculated separately, and associated hydraulic conductivity and porosity distributions at the cell center points.
- Multiple chemical species may be considered simultaneously
- Equilibrium-based, species-dependent sorption reactions with constant, spatially variable distribution (or retardation) coefficients may be considered.
- Radioactive decay chains may be specified (i) in a simple sense to represent simple decay or (ii) in a coupled sense to represent simultaneous decay and ingrowth along a specified multicomponent decay chain.
- A two-state,probabilistic, kinetic algorithm may be used to simulate matrix diffusion processes, kinetic attachment and detachment processes associated with colloidal transport or linear, kinetic sorption.

2. Installation, Use, and Code Organization

2.1 Installation

Unpacking

The SLIM-FAST source, example and documentation files may be untarred from the tar file into any directory. The path of this directory is needed later.

Building

SLIM-FAST may be built using the gnu `gfortran` compiler or the Intel, `ifort` compiler. To compile on Linux , Unix or OS-X using the gnu FORTRAN compiler there is a Makefile that also sits in the `/SLIM` directory. To compile the user can just type:

```
/SLIM_FAST> make
```

Installing tcl/tk

SLIM-FAST uses the scripting language tcl/tk to generate an input file and run the code. This is a free scripting language that is pre-installed on most systems. If needed it may be downloaded for PC, Linux/Unix or OS-X from: http://www.tcl.tk/

Running

The directory location for where slim sits must be specified in the environmental variable `SLIM_DIR`. An example is shown below:

Windows

```
set SLIM_DIR "f:/reed/SLIM "
```

Linux / OS-X

```
set SLIM_DIR "/home/maxwell/SLIM "
```

To run SLIM-FAST, the file `slim.run.tcl` is sourced from `$SLIM_DIR/tcl` directory, this is described in more detail in the next section and in the example.

2.2 Input Script Formats

As mentioned earlier, SLIM-FAST uses the scripting language tcl/tk to generate an input file and run the code. The following paragraphs describe the input components of the tcl/tk script.

Parameter file, run name, log file

Name the parameter file – this is the name of the file that will be generated by the tcl/tk script and read by the SLIM-FAST executable, it is a standard ascii input file where all the tcl variables are written for SLIM-FAST.

```
set parameter_file "file.par"
```

The following set the name of the run and the name of the log file.

```
set run_name  "Slim Run My Run"
set log_file  "slog.myrun.txt"
```

Domain Information

The variables nx, ny, and nz are the number of grid block in each direction, while dx, dy, and dz are the size of the grid blocks in length units that match those on your hydraulic conductivity. Currently, dx, dy and dz are constant.

```
set nx 50
set ny 50
set nz 50

set dx 10.
set dy 10.
set dz 1.
```

Groundwater Velocities

Groundwater velocities can either be read in, or calculated from conductivity, head, and porosity. Additionally, the velocity field can be steady-state or transient, There are three input cases to consider:

Case 1: Calculate steady state velocities:

```
set v_type calc
```

Read hydraulic conductivities in x,y and z

```
set kx_file "kx.pfb"
set ky_file "ky.pfb"
set kz_file "kz.pfb"
```

Read in head or pressure file

```
set press 0
```

```
set head_file "head.pfb"

set press 1
set head_file "pressure.pfb"
```

Specify or read in porosity

```
set phi_type "constant"
set phi 0.35

set phi_type "PFBFile"
set phi_file "porosity.pfb"
```

Specifications for saturation and relative permeability also need to set. SLIM-FAST will calculate the saturation and relative permeability using the vanGenutchen relationships. Additionally, a variable `saturated` set to "yes" may be used to specify fully-saturated conditions. For variably-saturated problems, the vanGenutchen α, n and residual saturation need to be specified.

Specify Saturated Conditions

```
set saturated "yes"
```

Set vanGenuchten paramters

```
set vga_file "vga.pfb"
set vgn_file "vgn.pfb"
set sres_file "sres.pfb"
```

Set timing (steady-state case). The units for time come from Kx,y,z. The variable `time_increment` determines the overall timestep dt, when concentration output will be recorded into an output file and when global particle functions (eg particle splitting) will be calculated. The variable `num_increments` is the number of timesteps over which the simulation will run. The settings below will result in a 365 (d) or 1 (y) timestep with total simulation time of 10 (y) and output every year.

```
set time_increment 365.
set num_increments 10
```

Case 2: Read in the velocities from a vbin file. The file contains time information and may be transient. All global timesteps (ie tnext) are governed by the information in this file.

```
set v_type vbin
set vel_file "v.bin"
```

The variable `vel_nskip` tells SLIM-FAST how many time steps to ignore from this file at the beginning of the simulation

```
set vel_nskip  0
```

Case 3: Calculated and transient velocity fields. The calculated, transient case works just like the steady-state case, except a number of times and pressure of head files are read in instead of a single file. These files and times are listed in a text file external to the input file.

```
set v_type "calc_trans"
set press 1
set head_list_file "press.txt"
set time_file "time.txt"
set num_increments 3

set v_type "calc_trans"
set press 0
set head_list_file "head.txt"
set time_file "time.txt"
set num_increments 3
```

Where the file time.txt and head.txt may look like:

File: `time.txt`

```
365.
730.
1460.
```

File: `press.txt`

```
Press1.pfb
Press2.pfb
Press3.pfb
```

Number of species or constituents

SLIM-FAST may solve over a number of different constituents, each with unique initial condition, sorption, attachment/matrix diffusion and decay properties. The variable `num_constituents` determines the number of constituents SLIM-FAST will look for in the variables for these processes, below.

```
set num_constituents 1
```

Decay/ingrowth

The variable `half_life(n)` where n is the number of constituents determines the exponential decay and ingrowth rate in units of (1/T) where time is specified by units of Kx,y,z. Constituents decay in a chain, with mass loss due to decay from 1 contributing to mass gain from constituent 2. The decay of the last constituent will contribute to the mass of constituent 1. It is important to specify a value of 0. (not very large values) if these processes are not desired.

```
set half_life(1) 10.
set half_life(2) 0.
```

Equilibrium Sorption or Chemical Retardation

Equilibrium chemical sorption or retardation can be specified in terms of (i) a constant linear retardation factor (R, type 1), (ii) a constant partition factor (kd, type 2), or (iii) a file-specified distribution of R or kd values. Different sorption values or even methods may be specified for each constituent.

```
set sorption_type(1) "constant"
set sorption_value(1) 1.0

set sorption_type(2) "constant"
set sorption_value(2) 2.0

set sorption_type(1) "R_file"
set R_file(1) "const1.R.pfb"
```

Attachment/Matrix Diffusion/Kinetic Sorption

Kinetic attachment and detachment parameters (1/T) which can be equivalent to kinetic sorption, which can be equivalent to some formulations of matrix diffusion (see Section 2) may be specified as constant or spatially variable, read in from a file. These parameters may be specified uniquely for each constituent. It is important to specify a value of 0. (*not* very large values) if these processes are not desired.

```
set attach_type(1) "constant"
set attachment_value(1) 0.
set detach_type(1) "constant"
set detachment_value(1) 0.

set attach_type(2) "constant"
set attachment_value(2)  4.62
set detach_type(2) "constant"
set detachment_value(2) 0.0869

set attach_type(1) "read-in"
set att_file(1) "katt.bin"
```

Concentrations

SLIM-FAST writes a concentration file in either a simple, binary format (.cnb) or vtk binary format for each constituent (M/L^3). For .cnb format, the variable `concentration_header(n)` is the beginning name for this file, with a number corresponding to timestep and file extension being appended to this name.

```
set write_concentrations "yes"
set concentration_header(1) "conc1"
set concentration_header(2) "conc2"
```

For VTK format the variable `write_concentrations` is set to "vtk" and the concentration header is written within the vtk file for each constituent (one file for all constituents) and a variable is needed for the overall filename, `concetration_filename`.

```
set write_concentrations "vtk"
set concentration_filename "conc
set concentration_header(1) "Metal"
set concentration_header(2) "Tracer"
```

Temporal averaging is a method to "trace" the particle over a gridblock by averaging particle mass by time of flight in that block. It may improve resolution for visualization of plumes but at a loss of concentration accuracy.

```
set temp_averaging "no"
set temp_averaging "yes"
```

Dispersion/Diffusion parameters

The variables `alpha_l` (L) and `alpha_t` (L) are longitudinal and transverse dispersivities. The variable `diffusivity` is molecular diffusivity (L^2).

```
set alpha_l 10.
set alpha_t 1.
set diffusivity 0.0
```

Wells

The number of wells, whether to print breakthrough and whether to overwrite or append the breakthrough file is specified.

```
set number_wells  1
set write_well_breakthrough "yes"
set write_well_breakthrough "no"
set well_overwrite "yes"
set well_overwrite "no"
```

The well file name is set by

```
set well_out_file(1) "well1.txt"
```

The well breakthrough delta time and number of steps, this is completely independent of all other time stepping in SLIM-FAST. Time units follow hydraulic conductivity. The below example specifies breakthrough be averaged over 30 (d) for a total time of 30,000 (d).

```
set well_breakthrough_dt 30.
set number_well_steps 1000
```

The location of each well in x, y and z over the well screen (L) is established by

```
set well_x_location(1) 50.
set well_y_location(1) 50.
set well_screen_top(1) 10.
set well_screen_bottom(1) 5.
```

The number in parenthesis after each variable designates the well number. The length units should match dx,y,z and Kx,y,z. SLIM-FAST will snap these to the nearest gridblock.

The well pumping rate (L^3/T). Units depend on units of Kx,y,z a value of zero denotes a monitoring well, where particles are not removed from the domain, just accounted for.

```
set well_pumping_rate(1) 0.0
set well_pumping_rate(1) 110.
```

Particle locations and spatial moments

Particle locations at each particle timestep may be written to a text file. This file can get humongous for even moderate numbers of particles.

```
set write_out_particles  "no"
set write_out_particles  "yes"
set particle_out_file  "parttrace.txt"
```

The spatial moments of a cloud or plume of particles can be calculated and written to a text file. These are computed ever global timestep.

```
set write_out_moments "no"
set write_out_moments "yes"
set moment_out_file  "moments.txt"
```

Forward/Backward simulation

Forward simulation moves particles forward along velocity vectors, backward simulation reverses the velocity direction (ie V=-V) but still simulates forward in time.

```
set simulation_mode "forward"
set simulation_mode "backward"
```

Particle Splitting

When dealing with low concentrations, normal variations in the number of "particles" of the solute have a great influence on the concentration. To reduce this numerical effect, below a certain concentration limit that you set (min_conc), SLIM-FAST can automatically split each particle into two particles each with half the mass of the first. This is done at global timestep

intervals (eg tnext) and the number of particles will grow to the value of npmax, described below.

```
set part_split "no"
set min_conc 1E-10
```

Breakthrough planes

SLIM-FAST will track the mass that crosses a plane that cuts the entire domain. The reporting interval is the same as that of the wells and must be specified there. In the plane location below, "x" designates the xy plane, "y" designates the yz plane, and "z" designates the xz plane. The array refers to the plane number, not constituent.

```
set number_planes  2
set plane_out_file(1) "plane1.txt"
set plane_xyz(1)  "x"
set plane_location(1) 50.

set plane_out_file(2) "plane2.txt"
set plane_xyz(2)  "x"
set plane_location(2) 60.
```

Initial Conditions

Particle initial conditions may be set as a "pulse" input or from a set of indictors specified in an external file. The input type may be set for each constituent, as designated by the number in parenthesis.

Pulse IC.

```
set slim_initial_condition_type(1) "pulse"
```

Location of the pulse IC: Particles are initialized randomly over an x,y,z coordinate range using the upper and lower values below. These are not snapped to the nearest grid block. The number in parenthesis refers to the constituent.

```
set slim_pulseIC_Xlower(1) 10.
set slim_pulseIC_Xupper(1) 20.
set slim_pulseIC_Ylower(1) 10.
set slim_pulseIC_Yupper(1) 20.
set slim_pulseIC_Zlower(1) 5.
set slim_pulseIC_Zupper(1) 6.
```

Initial concentration and number of particles: Particle mass and resolution is a function of these numbers and the volume of the source.

```
set slim_pulseIC_Initial_Concentration(1) 1.
set slim_pulseIC_number_particles(1) 100000
```

Source decay rate: This feature is not used in SLIM-FAST at this time. These values are merely placeholders.

```
set slim_pulseIC_decay_rate(1) 0.00001
set slim_pulseIC_decay_time(1) .000
set slim_pulseIC_decay_timesteps(1) 1
```

Multiple indicator IC: The following lines describe the multiple indicator initial condition type. This IC points to an ASCII text file that contains the information for this IC.

```
set slim_initial_condition_type(1) "ind_mult"
set slim_ic_ind_file(1) "ind.ic.txt"
```

An example indicator text file is shown below. The file needs a binary, indicator file that matches the SLIM-FAST computational grid and specifies the mass of particles that are assigned to given cells containing that indicator value. Particles are assigned as *mass per gridblock per given time interval*. Additionally, particle density per gridblock is specified in the file, so the total number of particles at the beginning of the simulation are determined by *num_timesteps x num_categories x particle_density x number_of_each_category*. So, for example, in the file below, if there were 10 nodes each of indicator 25 and 23 there would be 9,000 particles in the simulation (10x2x3x150). If the number of particles specified in this file exceeds the npmax variable set above the code will exit with a warning.

FILE: ind.ic.txt

```
!       Categorical IC - Example
!       Comment lines must be present as the code
!       ignores anything in these lines
!  c1 = 25; c2 = 23
"IC.ind.int"              ! category file
2                         ! number of categories, cats follow in order
25, 23
150, 150       ! particle density for each gridblock [Npart/Unit Mass]
3              ! number of timesteps
TS    Time Begin [T]    Time End [T]      Mass C1     Mass C2
1     0.00E+00          0.00E+00          9.10E-07    1.00E-06
2     1.00E+00          2.00E+01          5.19E-08    0.00E+00
3     2.00E+01          4.00E+01          8.48E-07    0.00E+00
```

Numerical Parameters

The following numerical parameters have an effect on how SLIM-FAST solves for particle movement. The variable `npmax` sets an upper limit for the number of particles for splitting or IC. This must be small enough to accommodate any memory constraints, but large enough to accommodate the IC or resolution requirements.

```
set npmax 750000
```

The parameter `give_up` is used to determine an upper limit to the number of particle steps a single particle may take in a global timestep (e.g., `tnext`). This is used to limit a single particle that may be stuck due to a mass-balance error in the flow field from slowing down the entire computation.

```
set give_up 1000
```

The parameter `epsilon` is used instead of zero in the numerical implementation of the particle solution. Many variables that have an absolute value less than epsilon will be assumed to be zero in the simulation.

```
set epsilon 1E-10
```

2.3 Executing SLIM-FAST

A tcl/tk source call to the file is the command that generates the parameter input file and executes SLIM-FAST. It may be called repeatedly from a single tcl script.

```
source $env(SLIM_DIR)/tcl/slim.run.tcl"
```

2.2 SLIM-FAST Code Structure

SLIM-FAST is organized into a number of subroutines. `slim_fast.f90` contains the primary of the code that is responsible for particle transport. `slim_f2.0.f90` handles most IO, problem setup and array allocation and calls the other subroutines. `v_calc.f90` calculates velocities for each timestep and `cbin_write.f90` writes the binary concentration output files; both are called in `slim_fast.f90`. These file structures are detailed in Section 2.4. The main array structure of SLIM-FAST is in the variable, `P(particle_number,particle_property)`, which is an allocatable double-precision array that contains all particle location, mass, constituent and time information. It is modified by `iP(particle_number,particle_property)`, which is an integer array of similar size that contains information regarding the state of a constituent that a particle may belong to. Both arrays are allocated by the variable `npmax`. The array `c(i,j,k,consituent)` is a single-precision allocatable array that contains the grid-based concentrations. The double-precision array `v(l,i*,j*,k*)` contains the cell-interface velocities for each orthogonal direction, where `i*` is equal to `i-1/2` for `l=1`, etc. If velocities are calculated, then head and hydraulic conductivity arrays are allocated and deallocated by `v_calc.f90`. A complete detail of the memory requirements for SLIM-FAST are given in Appendix B.6.

The main time loop in `slim_fast.f90` occurs over the number of timesteps at global time intervals that correspond to velocity updates and concentration output times. Within that loop, all particles are cycled over and moved from one global time to the next using many timesteps that are constantly adjusted for each particle. At the end of a global timestep concentrations are written and other output is written. Well and plane output is written at the end of the simulation.

2.4 SLIM-FAST File Types

Listed below are a number of file types used for input and output in SLIM-FAST. For each filetype the FORTRAN pseudo-code that SLIM-FAST uses is listed. It should be very straightforward for the user to modify this for their purposes. Most files in SLIM-FAST are binary format with a "BIG_ENDIAN" byte swap. For all the file types listed below, the following FORTRAN open statement format is used.

```
open(file,file=filename,form='unformatted',    &
recordtype='stream',convert='BIG_ENDIAN')
```

ParFlow binary files (.PFB) files

```
Real*8 X(nx,ny,nz), dx, dy, dz, X0, Y0, Z0
Integer*4 i,j,k, ix, iy, iz, ns,  rx, ry, rz, nx, ny, nz, nnx, nny, nnz, is

read(file) X0  ! x coordinate of lower left corner, not used
read(file) Y0  ! y coordinate of lower left corner, not used
read(file) Z0  ! z coordinate of lower left corner, not used
read(file) nx
read(file) ny
read(file) nz
read(file) dx
read(file) dy
read(file) dz
read(file) ns  ! number of subgrids, usually zero for serial output

do is = 0, (ns-1)

  read(file) ix    ! subgrid index starting location in x
  read(file) iy    ! subgrid index starting location in y
  read(file) iz    ! subgrid index starting location in z
  read(file) nnx  ! subgrid nx
  read(file) nny  ! subgrid ny
  read(file) nnz  ! subgrid nz

  read(file) rx   ! subgrid refinement, also not used by SLIM
  read(file) ry   ! subgrid refinement, also not used by SLIM
  read(file) rz   ! subgrid refinement, also not used by SLIM

  do  k=iz +1 , iz + nnz
    do  j=iy +1 , iy + nny
      do  i=ix +1 , ix + nnx
          read(file) x(i,j,k)
      end do
    end do
  end do

end do  ! is, subgrid loop
```

Concentration Binary (.CNB) files

```
INTEGER*4 nx,ny,nz,i,j,k,ijk
REAL*4 dx,dy,dz,x(nx,ny,nz)

WRITE(file) nx
WRITE(file) ny
WRITE(file) nz
WRITE(file) dx
WRITE(file) dy
WRITE(file) dz

DO  k=1,nx
  DO  j=1,ny
    DO  i=1,nz
      ijk = i + (j-1)*nx + (k-1)*nx*ny
      IF (x(i,j,k) > 0.d0) WRITE(file) ijk, x(i,j,k)
    END DO
  END DO
END DO
```

Binary Velocity (V.BIN) files

```
REAL*8  time, x(3,nx,ny,nz)
INTEGER*4 nx,ny,nz,i,j,k

READ(file) time
DO k = 1,  nz
  DO j = 1, ny
    DO i = 1, nx
      READ(file)  x(1,i,j,k),x(2,i,j,k),x(3,i,j,k)
    END DO
  END DO
END DO
```

Binary Indicator Files

```
INTEGER*4 nx,ny,nz,i,j,k, ic_indicator(nx,ny,nz)
READ(file) nx, ny, nz
  DO k = 1, nz
    DO j = 1, ny
      DO i = 1, nx
        READ(file) ic_indicator(i,j,k)
      END DO
    END DO
  END DO
```

3. Example Problem and Input Files

These examples are accompanied by a separate package of computer files, including the SLIM-FAST code and relevant scripts and input /output files.

3.1 Example: Cape Cod Tracer and Bacterial Transport

This is an example that describes a simulation of either a conservative bromide tracer or bacterial transport for the field experiment carried out by Ron Havey at the Cape Cod Site (Harvey and Garabedian, 1991). This example is courtesy of Jessica Lawson, jessica.lawson@jhu.edu. All the files to run this example are included in the SLIM-FAST tar/zip file.

SLIM-FAST runs from a tcl script. The program must of course know the geometry and geology of the domain, as well as properties of the "contaminant" and the location of detection wells and contamination source. It is the tcl script that provides the necessary info, or at least directs SLIM-FAST to the /flow directory generated by Parflow where the geology information can be found.

The tcl script for the Cape Cod project is called: `slim.harvey.example.tcl` and sits in the `$SLIM_DIR/test` directory.

When this script runs, it creates a new directory named `/slim.harvey` right in the directory where the tcl script is stored. SLIM-FAST then puts all its output in `/slim.harvey`. Of course, the user can change the name and location of this output directory by modifying `slim.harvey.example.tcl`.

SLIM-FAST will generate a set of concentration files (.cnb) that can be viewed later. Also a set of log files, a parameter file (which is useful if there are any problems, because this is exactly what is fed by the Tcl script to the Fortran program), an out file, and a set of well files describing the concentration at the wells you specify as a function of time.

To run SLIM-FAST,

- modify the tcl input script, if need be (remember to right-click and select edit in Windows or use your favorite text editor, VI, or such in OS-X/Linux)
- save the script (or save as, if you want to change the name)
- double click on the script to run it, or execute it as a shell script using `tclsh`

Now for the SLIM-FAST tcl input file:

```
#
#  Slim Fast Input Script
#
```

First set up a new directory for the results and open it. If you wanted to put an entire path rather than just a name, you would have more control over where your output file goes. For example, you would put

```
file mkdir "C:\\cape_cod\\revised_statistics\\slim.harvey"
```

and then cd into that directory using the path. You may of course name the directory anything you like. Note that you must use double slashes in the path; the single slash is a control character (in Windows).

```
file mkdir "slim.harvey"
cd "slim.harvey"
```

Then we set up SLIM-FAST to run on each one of the *n* flow realizations (in this case, 1, but we could easily loop over many realization).

```
set n_slim_runs 1

#
#  Loop through runs
#
for {set k 1} {$k <= $n_slim_runs} {incr k 1} {


#
#     run id and other info
#
```

Set up the parameter file – this is the name of the file that will be generated by the TCL script and read by SLIM-FAST. It is, essentially, the file that runs SLIM-FAST.

```
set parameter_file "slim.harvey.par"
```

An advantage of TCL is that filenames can have variables in them – like the increment variable k. The run name will be printed at the top of the parameter file. The log file records the run.

```
set run_name  "slim_harvey_brom R $k"
set log_file  "slog.harvey_brom.$k.txt"
```

SLIM-FAST needs velocities. The program can either read them in, or it can calculate them from permeability, head, and porosity (phi).

Case 1: Calculate velocities
We have to tell SLIM-FAST where to find the input files (which were output by the flow code ParFlow). ".." means "go up one directory." Note that these filenames also increment. SLIM-FAST will assume steady state flow. If you had your directory structure set up differently and going up one directory was not the appropriate action, you would need to state the whole path.

```
#
#     input files for head, perm and porosity
#
set v_type calc
set kx_file "../harvey_flow.$k.out.perm_x.pfb"
```

```
set ky_file "../harvey_flow.$k.out.perm_x.pfb"
set kz_file "../harvey_flow.$k.out.perm_x.pfb"
set press 0
set head_file "../harvey_flow.$k.head.pfb"
set saturated "yes"
```

Set the porosity.

```
set phi_type "constant"
set phi 0.39
```

Case 2: Read in the velocities.
First set the type, then tell SLIM-FAST where to find the velocity file. Again, ".." means "go up one directory." Reading in velocities allows you to deal with non-steady state flow conditions. In this version, Case 2 has been commented out , so SLIM-FAST is calculating velocities.

```
##set v_type vbin
set vel_file "../flow/harvey.$k.v.bin"
```

Set up your domain to match paflow in value and units. nx, ny, and nz are the number of grid block in each direction, while dx, dy, and dz are the size of the grid blocks in length units that match those on your hydraulic conductivity (in this case, meters).

```
#
#     grid and domain info
#

set nx 50
set ny 30
set nz 100
set dx .34
set dy .34
set dz .038

# particle number and numerics
set npmax 750000
set give_up 1000
set epsilon 1E-10
```

In this case, we have only one thing of interest in our system: the conservative tracer bromide. You could also have other tracers, etc that you were modeling at the same time, which would increase the number of constituents.

```
#
# number of constituents
#
set num_constituents 1
```

Bromide doesn't decay. The variable half_life(n) describes exponential decay AND ingrowth, and the subscript n allows you to set different half lives for different constituents if you have a multi-constituent system.

```
#
```

```
# radioactive decay for constituents
#
set half_life(1) 0.
set half_life(2) 0.
```

Set transport-related parameters. alpha_l and alpha_t are longitudinal and transverse dispersivities. Diffusivity is molecular diffusivity.

```
#
#     Dispersion and Diffusion Parameters
#
set alpha_l 0.00
set alpha_t 0.00
set diffusivity 0.0
#
```

Sorption (i.e. chemical retardation, which is not kinetic) can be constant (for a conservative tracer), as is shown here. There could also be a linear relation between ln(K) and the rate R (in which case you would write, eg: set R_a -1). You could also read in values from a file. You can set different sorption values for each constituent.

```
#     set linear sorption parameters
#
set sorption_type(1) "constant"
set sorption_value(1) 1.0

set sorption_type(2) "constant"
set sorption_value(2) 1.0
```

Here is where you set your attachment and detachment parameters. It is the final statement that counts as your setting. You can set diffusion parameters for each constituent.

```
#
#     set matrix diffusion or Att/det parameters
#
set attach_type(1) "constant"
set attachment_value(1)  [expr (66. * 0.007) ]
set attachment_value(1)  4.62
set attachment_value(1) 0.
set detach_type(1) "constant"
set detachment_value(1) 0.0869
set detachment_value(1) 0.00

set attach_type(2) "constant"
set attachment_value(2)  [expr (66. * 0.007) ]
set attachment_value(2)  4.62
set attachment_value(2) 0.
set detach_type(2) "constant"
set detachment_value(2) 0.0869
set detachment_value(2) 0.00
```

The units for time come from K (in this case, days). time_increment 4 means that concentration output will be recorded into an output file for every 4th day. num_increments 10 means that the simulation will run for 10x4 or 40 days.

```
#
#     Set Run Timing Info
#
set time_increment 4.
set num_increments 10
```

If you are reading in velocities from a file, this next line tells the program how many values to skip between readings – i.e. if you have a velocity recorded every 1 day, setting vel_skip to 2 would mean that SLIM-FAST only reads every other velocity value – one every 2 days. If you have set SLIM-FAST to calculate velocities, this line will be ignored.

```
set vel_nskip  0
```

You tell SLIM-FAST that you want it to print the breakthrough (concentration) data for each well.

```
#
#     SlimNG Output
#
set write_well_breakthrough "yes"
set well_overwrite "yes"
```

You name the file the breakthrough data will be recorded in. You will want a different output file for each constituent if you have a multi-constituent system.

```
set well_out_file(1) "well.EX.$k.txt"
set well_out_file(2) "well.EX2.$k.txt"
```

SLIM-FAST will report a concentration value every day for 40 days.

```
set well_breakthrough_dt 1.
set number_well_steps 40
```

SLIM-FAST will also write a plume file, which is something like a spatial snapshot. It will do this every time interval (i.e. every 4th day) for the duration of the run (40 days). Different filenames for different constituents.

```
set write_concentrations "yes"
set concentration_header(1) "conc_brom.$k"
set concentration_header(2) "conc_broml.$k"
```

Alternately, you may also specify VTK output

```
set write_concentrations "vtk"
set concentration_filename "conc.$k"
set concentration_header(1) "Bromide"
set concentration_header(2) "PRD1"
```

A number of other output options… we are not using them.

```
set write_out_particles  "no"
set particle_out_file  "parttrace.txt"
set write_out_moments "no"
set moment_out_file  "moments.txt"
set temp_averaging "no"
```

If you are tracing well data back to its source, you want to run SLIM-FAST "backward." If you are eg. contaminating a well and following the plume, you want to run SLIM-FAST "forward." For Cape Cod, we know our source and are observing breakthrough, so we will run the model forward.

```
#
#     mode of operation
#   expects "forward" or "backward"
set simulation_mode "forward"
```

When dealing with low concentrations, normal variations in the number of "particles" of the solute have a great influence on the concentration. To reduce this numerical effect, below a certain concentration limit that you set (min_conc), SLIM-FAST can automatically split each particle into two particles each with half the mass of the first.

```
set part_split "no"
set min_conc 1E-10
```

Temporal averaging is another way to increase resolution at low concentrations. It is good for visualization, but not if you are interested in absolute concentrations.

```
set temp_averaging "no"
```

Tell SLIM-FAST how many wells to expect.

```
#
#     well information
#
#
set number_wells  2
```

Set the location of each well in x and y. The number in parenthesis (n) after each variable designates the well number. The model grid has its own origin at (0,0,0) and these locations, as with all locations in SLIM-FAST, need to be defined using the model grid. The well will snap to the grid, and SLIM-FAST will assume that the well is over 1 grid block in x and y.

```
# monitoring port on M01 at 8.510 m bls
set well_x_location(1) 12.07
set well_y_location(1) 5.27
```

The sampling port on the well is assumed to have some height in z. The `well_screen_xxxx` variables allow you to set the positions of the top and bottom of the sampling port.

```
set well_screen_top(1) 1.9
set well_screen_bottom(1) 1.862
```

If you are pumping, set the pumping rate. Units depend on units of K (in this case m^3 / day). You would also have had to set up your pumping wells in your flow code (in this case ParFlow), and these, being the same wells, would have to be set the same way.

```
set well_pumping_rate(1) 0.000
```

Do the same for the rest of the wells.

```
# monitoring port on M01 at 9.089 m bls
set well_x_location(2) 12.07
set well_y_location(2) 5.27
set well_screen_top(2) 1.330
set well_screen_bottom(2) 1.292
set well_pumping_rate(2) 0.000
```

You can also specify a plane and SLIM-FAST will give you output describing how much mass crosses that plane. The reporting interval will be the same as that of the wells. "x" designates the xy plane, "y" designates the yz plane, and "z" designates the xz plane. In this case, the number in parenthesis refers to the plane, if you have more than one. Since we have zero planes, none will be reported, and the following lines will be ignored. I have left them here as a reference for the proper syntax of the commands.

```
#
#     Plane Information
#
set number_planes  0
set plane_out_file(1) "plane.brom.$k.txt"
set plane_xyz(1)  "x"
set plane_location(1) 11.9

set plane_out_file(2) "plane.brom.$k.txt"
set plane_xyz(2)  "x"
set plane_location(2) 11.9
```

Here is where you set your initial conditions. A "pulse" input is appropriate for a single injection over a time period that is short compared to the duration of the run (i.e. 1-2hrs over 40 days). Set the input type for each constituent, as designated by the number in parenthesis.

```
#
#     Slim NG Initial Condition
#
# valid types are:
# ind-mulitple
# indicator file
# pulse
set slim_initial_condition_type(1) "pulse"
set slim_initial_condition_type(2) "pulse"
```

SLIM-FAST will calculate the true particle mass according to the initial concentration, volume of the source, and number of particles you specify.

Set the location of the initial pulse. You have control of the precise upper and lower limits of the pulse location in all three dimensions; it does not snap to a grid block. The number in parenthesis refers to the constituent.

```
set slim_pulseIC_Xlower(1) 4.76
set slim_pulseIC_Xupper(1) 5.525
set slim_pulseIC_Ylower(1) 5.1
set slim_pulseIC_Yupper(1) 5.44
set slim_pulseIC_Zlower(1) 1.1
set slim_pulseIC_Zupper(1) 2.1
```

If you set the initial concentration to 1, as we have here, then all concentrations calculated throughout the program will be as if normalized. However, you may set it to whatever value you like. The initial number of particles must be high enough that your solute will be detectable when diluted downgradient.

```
set slim_pulseIC_Initial_Concentration(1) 1.
set slim_pulseIC_number_particles(1) 100000
```

This decay rate describes decay of the source term over time, and NOT decay of the particles. It is an exponential and can't be set to zero, but the following lines are satisfactory for an "instant" injection.

```
set slim_pulseIC_decay_rate(1) 0.00001
set slim_pulseIC_decay_time(1) .000
set slim_pulseIC_decay_timesteps(1) 1
```

And repeat the initial conditions for the second constituent, if you have one. If you don't, no need to delete them, they will just be ignored.

```
set slim_pulseIC_Xlower(2) 4.1
set slim_pulseIC_Xupper(2) 4.44
set slim_pulseIC_Ylower(2) 5.5
set slim_pulseIC_Yupper(2) 5.84
set slim_pulseIC_Zlower(2) 1.2
set slim_pulseIC_Zupper(2) 1.6
set slim_pulseIC_Initial_Concentration(2) 1.
set slim_pulseIC_number_particles(2) 100000
set slim_pulseIC_decay_rate(2) 0.00001
set slim_pulseIC_decay_time(2) .000
set slim_pulseIC_decay_timesteps(2) 1
```

Finally, this is the line that actually runs slim. The path must be correct, of course.

```
source "C:\\APPS\\SLIMFAST\\ slimfast_5-11-05.run.tcl"
```

The final parenthesis closes the do loop for all realizations.

```
}
```

If you have trouble, it may be helpful to look at the parameter file to see exactly what you fed to the Fortran code.

You will be able to view some of your results while the simulation is running. The concentration plume files (.cnb) that show the plume at each of the reporting intervals you designated can be loaded by a visualization program and projected onto the geology.

4. References

Abulaban, A., and J. L. Nieber, 2000, Modeling the Effects of Nonlinear Equilibrium Sorption on the Transport of Solute Plumes in Saturated Heterogeneous Porous Media, *Advances in Water Resources* 23(8), 893–905

Ahlstrom, S., H. Foote, R. Arnett, C. Cole, and R. Serne, 1977, Multicomponent Mass Transport Model: Theory and numerical implementation, Battelle Pacific Northwest Laboratories, Richland, WA, BNWL 2127

Carle. S. F., R. M. Maxwell, and G.A. Pawloski, 2003. Impact of Test Heat on Groundwater Flow at Pahute Mesa, Nevada Test Site. Lawrence Livermore National Laboratory, Livermore California (UCRL-ID-152599)

Friedlander, G., J. Kennedy, E. Macias, and J. Miller, 1981, *Nuclear and Radiochemistry*, 3rd ed., John Wiley and Sons, New York, NY.

Gelhar, L. W. (1993), *Stochastic Subsurface Hydrology*, Prentice Hall.

Harvey, R. W. and S. P. Garabedian, 1991. Use of colloid filtration theory in modeling movement of bacteria through a contaminated sandy aquifer. *Environmental Science and Technology* 25, 178-185.

Kinzelbach W, 1988, The random walk method in pollutant transport simulation, in Groundwater Flow and Quality Modeling, NATO ASI Ser., Ser. C Math and Phys. Sci., vol. 224, edited by E. Custido, A. Gurgui, and J.P. Lobo Ferreria, pp. 227-246, D. Reidel, Norwell, Mass

LaBolle, E. M., G. E. Fogg, and A. F. B. Tompson, 1996. Random-walk simulation of transport in heterogeneous porous media: Local mass-conservation problem and implementation methods, *Water Resources Research,* 32(3), 583-593

Liu, H. H., G. S. Bovarsson, and L. Pan (2000), Determination of Particle Transfer in Random Walk Particle Methods for Fractured Porous Media, *Water Resources Research* 36(3), 707–713

Maxwell, R.M. and W.E. Kastenberg, A Model for Assessing and Managing the Risks of Environmental Lead Emissions. *Stochastic Environmental Research and Risk Assessment*, 13(4), 231-250, 1999.

Maxwell, R.M., 1998. Understanding The Effects Of Uncertainty And Variability On Groundwater-Driven Health Risk Assessment. Ph.D. Dissertation, University of California, Berkeley,

Maxwell, R.M. and W.E. Kastenberg, and Y. Rubin, 1999. A methodology to integrate site characterization information into groundwater-driven health risk assessment, *Water Resources Research,* 35 (9): 2841-2855

Pawloski, G.A., A.F.B. Tompson, and S.F. Carle, eds., 2001. Evaluation of the hydrologic source term from underground nuclear tests on Pahute Mesa at the Nevada Test Site: The CHESHIRE test, Lawrence Livermore National Laboratory, Livermore, CA (UCRL-ID-147023)

Rubin, Y., A. Sun, R.M. Maxwell, and A. Bellin, 1999. The concept of block effective macrodispersivity and a unified approach for grid-scale and plume-scale dependent transport, *Journal of Fluid Mechanics*, 395, 161-180

Maxwell, R. M., C. W. Welty, and A. F. B. Tompson (2003), Streamline-Based Simulation of Virus Transport Resulting from Long Term Artificial Recharge in a Heterogeneous Aquifer, *Advances in Water Resources* 26(10), 1075-1096.

Schafer-Perini, A.L. and J.L. Wilson, Efficient and accurate front tracking for two-dimensional groundwater flow models, *Water Resources Research.*, 27(7), 1471-1485, 1991.

Steefel, C. I. and S. B. Yabusaki, 1996, OS3D/GIMRT, Software for Modeling Multicomponent and Multidimensional Reactive Transport, User manual and *programmer's guide, Version 1.0*, Pacific Northwest National Laboratory, Richland, WA, PNL-11166.

Tompson, A.F.B., E. G. Vomoris, and L.W. Gelhar, 1987. Numerical simulation of solute transport in randomly heterogeneous porous media: Motivation, model development, and application, Lawrence Livermore Natl. Lab., Livermore, Calif. (UCUD-21821)

Tompson A.F.B. and L.W. Gelhar, 1990. Numerical simulation of solute transport in three-dimensional randomly heterogeneous porous media. *Water Resources Research*, 26(10), 2541-2562.

Tompson A.F.B, 1993. Numerical simulation of chemical migration in physically and chemically heterogeneous porous media. *Water Resources Research*, 29(11), 3709-3726.

Tompson, A.F.B. and D. E. Dougherty, 1992. Particle-grid methods for reacting flows in porous media with application to Fisher's equation. *Applied Mathematical Modelling.*, 16, 374-383.

Tompson, A. F. B., A. L. Schafer, and R. W. Smith, 1996, Impacts of physical and chemical heterogeneity on co-contaminant transport in a sandy porous medium, *Water Resources Research*, 32(4), 801-818.

Tompson, A.F.B., R.D. Falgout, R. D., S.G. Smith, W.J. Bosl, and S.F. Ashby, 1998, Analysis of subsurface contaminant migration and remediation using high performance computing: *Advances in Water Resources*, v. 22, n. 3, p. 203-221

Tompson, A. F. B., S. F. Carle, N. D. Rosenberg, and R. M. Maxwell, 1999, Analysis of groundwater migration from artificial recharge in a large urban aquifer: A simulation perspective, *Water Resources Research*, 35(10), 2981-2998.

Tompson, A. F. B., R. M. Maxwell, S. F. Carle, M. Zavarin, G. A. Pawloski, and D. E. Shumaker, 2005. Evaluation of the Non-Transient Hydrologic Source Term from the

CAMBRIC Underground Nuclear Test in Frenchman Flat, Nevada Test Site, Lawrence Livermore National Laboratory, Livermore, CA (UCRL-TR-217191)

Uffink G.J.M, 1988. Modeling of solute transport with random walk method, in Groundwater Flow and Quality Modeling, NATO ASI Ser., Series C Math and Physical Sciences, vol. 224, edited by E. Custido, A. Gurgui, and J.P. Lobo Ferreria, pp. 247-265, D. Reidel, Norwell, Mass.

Zavarin, M., S.F. Carle, and R.M. Maxwell, 2004. Upscaling Radionuclide Retardation-Linking the Surface Complexation and Ion Exchange Mechanistic Approach to a Linear Kd Approach, Lawrence Livermore National Laboratory, Livermore, CA (UCRL-TR-204713)

Appendix A: Mathematical Model of Mass Transport

A.1 General Conceptualization and Approach

The mathematical models for chemical transport used in SLIM-FAST are based primarily on a single continuum conceptualization for chemical transport in a saturated porous medium. In this approach, both fluid flow and chemical transport processes are assumed to occur either (i) within the interstices of a porous and unfractured medium, such as in unconsolidated alluvium or sedimentary formation or (ii) within the primary fractures of a fractured rock network. In the latter case, SLIM-FAST can also handle a limited dual continuum formalism that allows for diffusive transport (but no fluid flow) into and out of porous matrix blocks that separate the primary flow fractures. The fractured rock or porous medium variables, such as aqueous concentration, groundwater velocity, and medium porosity, are assumed to be defined over appropriate and compatible representative elementary volumes (REVs) of the medium, and balance equations written for mass, momentum, or energy quantities in each porous continuum will share appropriate terms for exchange of these quantities.

SLIM-FAST is designed (only) to simulate transient mass transport processes within a porous or fractured rock medium. Steady state or transient groundwater flow fields, as derived from other numerical or analytic models, need to be supplied externally. As such, feedback or coupling to a corresponding flow model, as may occur if there is some concentration-dependent flow effect, cannot be treated.

A.2 Mass Transport Equations

In general, the balance equations for chemical mass and reactive transport in a porous medium will reflect physical advection (flow) processes, dispersion and molecular diffusion effects, recharge or extraction mechanisms (e.g., in wells), and a number of specific geochemical reaction phenomena related to, for example, aqueous complexation, radioactive decay or ingrowth, precipitation or dissolution, ion exchange and surface complexation between dissolved species and reactive minerals in the porous medium.

Within SLIM-FAST, these processes are simplified to yield a mass transport equation valid for dilute solutions where

- The collection of species is neutrally buoyant
- Sorption (ion exchange/surface complexation) reactions are assumed to be in instantaneous equilibrium, and
- Aqueous complexing reactions are not considered.

Under these conditions, the mass balance for species *j* takes the form:

$$\frac{\partial}{\partial t}\left(\phi\left(c_j + c_j^{im}\right)\right) + \nabla \bullet \left(\phi \mathbf{v} c_j\right) - \nabla \bullet \left(\phi \mathbf{D} \bullet \nabla c_j\right) =$$

$$-\lambda_j \phi\left(c_j + c_j^{im}\right) + \lambda_k \phi\left(c_k + c_k^{im}\right)$$

$$-R_j^{\min} - s_j^{fm} - c_j \sum_w Q_w \delta(x - x_w)\delta(y - y_w)\delta(z - z_w)$$

(A1)

where

- c_j represents the aqueous concentration of primary species *j* in solution (M-aqueous/L^3-aqueous)
- c_j^{im} represents the immobile concentration of primary species *j* sorbed onto the solid phase (M-sorbed/L^3-aqueous)
- $c_j^T = \phi(c_j + c_j^{im})$ represents the total (bulk) concentration (M-aqueous plus M-sorbed/L^3-bulk) of species *j*,
- ϕ and $\mathbf{v}$ represent the meduim porosity (-) and groundwater flow velocity (L/T),
- $\mathbf{D}(\mathbf{x}) = (\alpha_T V + D_e)\mathbf{I} + (\alpha_L - \alpha_T)\frac{\mathbf{vv}}{V}$ is the hydrodynamic dispersion tensor (L^2/T), where α_L and α_T are longitudinal and transverse medium dispersivities (L) and D_e is an effective molecular diffusivity (L^2/T) for the porous medium,
- λ_j represents the (radioactive) decay rate of species j (1/T), also equal to $\ln(2)/t_{1/2}$, where $t_{1/2}$ is the half life of species j (T),
- $R_j^{\min}$ represents the rate of loss or gain of aqueous mass from mineral dissolution or precipitation reactions (M/T)
- s_j^{fm} represents the rate of loss or gain of aqueous mass in a fracture regime to and from the matrix regime from matrix diffusion (M/T), and
- Q_w represents the volumetric rate of pumping (fluid loss, L^3/T) from a well at location (x_w, y_w, z_w).

The quantity $c_j^T = \phi(c_j + c_j^{im})$ represents the total mass of species *j*, aqueous or sorbed, per unit bulk volume of the medium. It should be distinguished from the sum $c_j + c_j^{im}$, which denotes the total mass, aqueous or sorbed, per unit volume of solution or pore volume. The ratio $R_j = 1 + c_j^{im}/c_j = c_j^T/\phi c_j$ represents the instantaneous state of partitioning between the aqueous and immobile fractions of species *j* at a point in space and time. Thus, with no loss of generality, (A1) may also be rewritten as

$$\frac{\partial c_j^T}{\partial t} + \nabla \bullet \left(\frac{\mathbf{v} c_j^T}{R_j} \right) - \nabla \bullet \left(\phi \mathbf{D} \bullet \nabla \frac{c_j^T}{\phi R_j} \right) = -\lambda_j c_j^T + \lambda_k c_k^T$$

$$- R_j^{\min} - s_j^{fm} - \frac{c_j^T}{\phi R_j} \sum_w Q_w \delta(x - x_w) \delta(y - y_w) \delta(z - z_w) \qquad \text{(A2)}$$

In general, R_j may be a complicated function of c_j, the mineralogic characteristics of the porous medium, and the concentrations of aqueous species different from j including other species, natural aqueous minerals, or the pH (Tompson, 1993; Tompson et al., 1996). However, under sufficiently dilute conditions, R_j may become independent of one or all of the species concentrations or the pH, such that, in the most convenient case, it reduces to a simple medium property. In this case, R_j is often referred to as a retardation coefficient, and the partitioning among the mobile and immobile fractions of species j can be described by a single medium-dependent constant. Alternatively, the retardation factor is expressed by

$$R_j \approx 1 + \frac{\rho_b K_{d,j}}{\phi} \qquad \text{(A3)}$$

where $K_{d,j}$ is the so-called equilibrium distribution coefficient [L^3/M], and ρ_b is the bulk density of the porous medium [M/L^3].

The use of retardation coefficients to describe partitioning behavior in this way is an approximate, continuum scale concept that should ideally be based on or confirmed with fundamental, process-based descriptions of water rock interactions that occur at the pore or mineral surface scale. It is an approximation that may be viable in some scenarios and a totally inadequate one in others.

When the retardation factor approach is valid and mineralogic conditions are constant in space, the retardation effect will be constant in space such that the migration rate of the aqueous species is uniformly reduced by a factor R_j relative to the groundwater velocity. If the mineralogic conditions are variable in space, then the retardation effect will be spatially variable. This may lead to an enhanced dispersion effect (Gelhar, 1993), owing to the fact that the migration rates of different radionuclide parcels may be more widely distributed as a result. There is some evidence that R_j or $K_{d,j}$ may be correlated to the natural log of hydraulic conductivity. A simple model of correlation can be included in SLIM-FAST [Tompson, 1993]

$$\phi K_{d,j} \approx k_1 + k_2 \ln K(\mathbf{x}) \qquad \text{(A4)}$$

where parameters k_1 and k_2 depend upon site specific features. It is important to recognize that this correlation is approximate; chemical- and site-specific geological features may result in different models of sorption behavior.

Appendix B: Particle Transport Algorithm

B.1 Introduction

Particle-based transport models have been widely applied to solve problems in hydrology, atmospheric sciences, fluid mechanics, plasma dynamics, astrophysics, and many other types of mass, momentum, and energy transport applications. Recent investigations have shown that these techniques can be effectively and efficiently applied to solve equations describing conservative and reactive chemical transport in subsurface porous formations (Ahlstrom et al., 1977; Kinzelbach, 1988; Uffink, 1988, Tompson and Gelhar, 1990; Tompson and Dougherty, 1992; Tompson, 1993; Tompson, et al., 1996; Maxwell and Kastenberg, 1999; Abulaban and Nieber, 2000; Liu et al., 2000). They suffer little from negativity, numerical dispersion, or mass balance problems seen in many grid-oriented approaches at large grid-Peclet numbers.

The particle model employed in SLIM-FAST is based upon a simplified version of the total mass balance equation (2) in which the physical and geochemical processes are simplified to the extent that:

- Ion exchange and surface complexation reactions are described by simple retardation coefficients that can be considered functions or properties of the local mineralogy and ambient solution composition.
- Precipitation and dissolution of secondary minerals is ignored.
- The pH and groundwater composition are assumed constant.
- Matrix diffusion is accounted for in an approximate manner.

B.2 SLIM-FAST Particle Model

In the current particle approach, the spatial distribution of total species mass, as represented by the total concentration c_j^T, is approximated by a finite system of N_j particles,

$$c_j^T(\mathbf{x},t) = \phi\left(c_j + c_j^{im}\right) = \phi R_j c_j = \sum_{p=1}^{N_j} m_p \delta\left(\mathbf{x} - \mathbf{X}_p(t)\right) \tag{B1}$$

where δ is a Dirac function and R_j is the retardation factor associated with species j. The particles may carry, or be associated with, different species attributes such as mass (m_p), position ($\mathbf{X}_p$), type (j), age ($t - t_0$), or even its regime or phase of existence (e.g., fracture, matrix).

Although the total concentration is considered to be a spatially continuous function, its particle representation in (B1) is discontinuous and may be difficult to graphically portray. To aid in the visualization process, the masses and positions of particles associated with radionuclide j may be used at any time to estimate c_j^T at any point $\mathbf{x}$ via (B1) or a modified expression of the form

$$c_j^T(\mathbf{x},t) = \phi\left(c_j + c_j^{im}\right) = \phi R_j c_j = \sum_{p=1}^{N_j} m_p \zeta\left(\mathbf{x} - \mathbf{X}_p(t)\right) \tag{B2}$$

In this representation, the function ζ is used to project the contributions of particles near the point **x** to the value of c_j^T at **x** and provide a smoother solution (Bagtzoglou et al., 1992). For example, in the applications below, $\zeta(\mathbf{x} - \mathbf{X}_p)$ is defined to be $1/V_b$ when $\mathbf{X}_p$ lies within a grid block of volume V_b surrounding the point **x**, and zero otherwise. Because the total concentration is the fundamental variable in this representation, the corresponding aqueous and sorbed fractions of c_j^T at a point must be determined by decomposing a computed value of c_j^T with known values of ϕ and R_j at that point.

A simulation is initialized by mapping specified distributions of c_j^T and other relevant attributes onto a field of particles in a manner consistent with (B1) or (B2). The number of particles used to represent a unit of total species mass is defined as the particle resolution, N_r, and may be controlled to improve the quality of the solution. The simulation proceeds over discrete time steps by changing the various particle attributes. This will involve, for example,

- Moving the particles according to a known background velocity field and other medium characteristics associated with dispersion and chemical retardation forces.
- Changing the type of each particle to represent radioactive decay from one radionuclide species (j) into another (k).

As posed here, the particle model is inherently mass conservative in the sense that

- The total mass between the aqueous and sorbed states is conserved.
- The total mass within a radionuclide decay chain is conserved.
- The total mass moving between the fracture and matrix regimes is conserved.
- These features comprise the most attractive attributes of the particle model.

B.3 Advection, Dispersion, and Retardation Processes

The movement of particles is based upon an explicit random walk algorithm (Kinzelbach, 1988; Uffink, 1988, Tompson and Gelhar, 1990; Tompson, 1993; Maxwell and Kastenberg, 1999; Abulaban and Nieber, 2000),

$$\mathbf{X}_p(t + \Delta t) = \mathbf{X}_p(t) + \left(\frac{\mathbf{v}}{R_j} + \frac{\nabla \bullet \mathbf{D}}{R_j} + \frac{\mathbf{D} \bullet \nabla(\ln \phi)}{R_j}\right)\Delta t + \mathbf{B}_j \bullet \mathbf{Z}\sqrt{\Delta t} \tag{B3}$$

In this expression, the second term on the right accounts for particle displacement along flow streamlines and includes two factors to correct for nonuniform distributions of ϕ or **D**. The porosity correction is usually a small quantity and is typically neglected. The third (random walk) term on the right accounts for the dispersive flux, where $\mathbf{B} \bullet \mathbf{B}^T = 2\mathbf{D}/R_j$ and **Z** is a random vector whose independent components have zero mean and unit variance. Ultimately, the particle mass density evolved through repeated use of (B3) on all particles will satisfy a conservative

(e.g., zero right-hand side) form of the simplified mass balance equation (B2) in the limit as N_p or $N_r \to \infty$ (Tompson and Gelhar, 1990).

The time step

The time step, Δt, used in this algorithm is chosen uniquely for each individual particle as a function of accuracy limits imposed by the velocity field, in conjunction with other limits associated with the dispersion, matrix diffusion, and radioactive decay steps discussed both here and later.[1] Typically, provisional values are chosen with respect to the constraints associated with each process, with the lowest value ultimately being selected for use. For example, the provisional time step associated with the displacement equation (D3) is selected for each particle to advance it via advection along an interpolated streamline within each grid block, at most from one edge to the next (Pollock, 1988; Schafer-Perini and Wilson, 1991), but potentially over shorter distances if a change in the velocity field or constraints from other mechanisms dictate. The value will be modified to account for retardation if $R_j > 1$.

The advective substep (see below) is followed by other substeps (see below) that address the correction and random portions of the displacement shown in equation (D3). Provisional time steps are chosen independently for these steps as well, each limited by the magnitude of the velocity gradient or the largest dispersion coefficient.

Because different time steps are used for different particles, periodic rendezvous times, *T*, may be identified to collect the position and state of all particles for visualization and other interrogation purposes (Tompson et al., 1988; Maxwell, 1998; Maxwell and Kastenberg, 1999).

Advection algorithms

`slim-fast.f` was written specifically to exploit a quasi-analytical formulation described in Shafer-Perini and Wilson (1991). In this paper, a rapid solution method for the advection terms in Equation (B3) was presented. Analytical techniques are used to solve for particle velocity streamlines in each grid block, based on linear interpolation of boundary fluxes. Figure B1 shows a subset of a 3D seven-point computational grid (in the x, y plane), centered on block or cell *(i, j, k)*.

[1] This differs from the use of a uniform time step that may be used to advance the position of all particles simultaneously (e.g., as in Tompson and Gelhar, 1990).

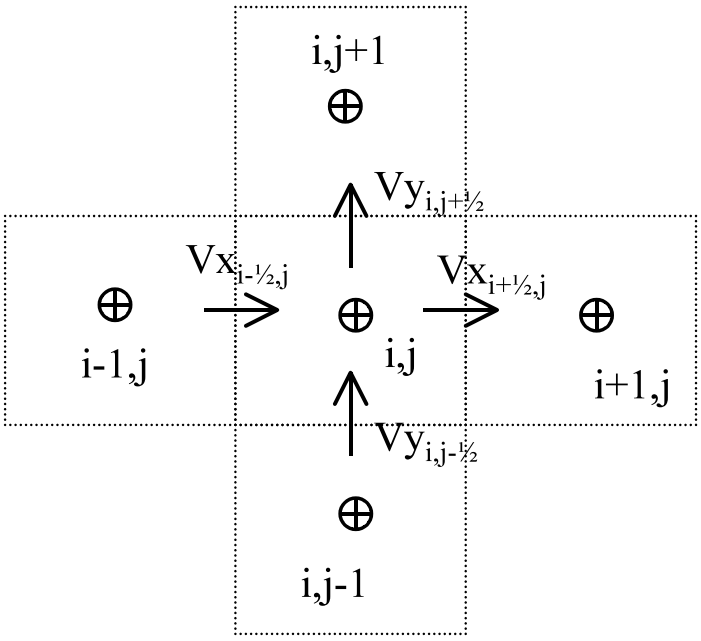


Figure B1. Five cells of a 3D, seven-point finite difference stencil with x, y cell velocity components shown in the x, y plane

At all grid cell center locations, values for the hydraulic conductivity ($K_{i,j,k}$) and hydraulic head ($h_{i,j,k}$) are specified from external flow simulations and other parametric specifications. The following equation is used to calculate the groundwater flow velocity for the *x* direction across the *i, i+1* block face:

$$v_x\left(i+\tfrac{1}{2},j,k\right)=K_{j,j,k}\left[\frac{2}{1/K_{i+1,k,j}+1/K_{i,k,j}}\right]\bullet\left[\frac{h_{i+1,j,k}-h_{i,j,k}}{\phi\Delta x}\right] \tag{B4}$$

where ϕ is the porosity, Δx is the spatial discretization of in the *x*-direction, $K_{i,j,k}$ is the hydraulic conductivity at cell *(i, j, k*), and $h_{i,j,k}$ is the head potential calculated at cell *(i, j, k).* Similar equations are used to calculate velocities in the y and z directions. We can now write an expression for the velocity anywhere in the cell (denoted V_{xP} to signify velocity at the exact location of the particle of interest) as

$$v_{xP}=a+b(x-x_0) \tag{B5}$$

and the constants *a* and *b* may be written as

$$a=v_x\left(i-\tfrac{1}{2},j,k\right) \tag{B6}$$

and

$$b=\frac{v_x\left(i+\tfrac{1}{2},j,k\right)-v_x\left(i-\tfrac{1}{2},j,k\right)}{\Delta x}. \tag{B7}$$

Where the location x_0 corresponds to the leftmost location of the current grid cell. We now write the Darcy velocities as time derivatives of fluid displacement

$$v_{xP} = a + bx = \frac{dx}{dt}. \tag{B8}$$

It is possible to integrate this function and relate the particle position (shown as two x locations, x_0 and x_1) to the natural log of time as

$$\Delta t = \frac{1}{b}\ln\left(\frac{a + bx_1}{a + bx_0}\right) \tag{B9}$$

Special care must taken when applying this methodology to screen for cases when (B9) is singular. A common case is a zero slope (resulting from a uniform velocity in the cell) in which Equation (B9) simplifies to

$$\Delta t = \frac{(x_1 - x_0)}{a}. \tag{B10}$$

The equation to describe a particle's displacement along an interpolated streamline is as follows:

$$x = \frac{(a + bx_0)e^{b\Delta t} - a}{b} \tag{B11}$$

As mentioned above, there are two choices for particle movement: (i) Δt may be assigned, and the particle moved to a corresponding location, or (ii) Δt may be calculated based on a desired location. A mixture of these two methods is employed in `slim-fast.f`. The scheme used is based on each particle's movements being independent of other particles and works as follows.

1. A particle starts at some initial location in the domain, x_0 at some time t_0.
2. Some later time, t_1, is specified, which has a corresponding unknown particle location, x_1.
3. The particle is advected from its initial location to the location of the nearest cell boundary. This is accomplished by calculating the time to reach every side of the cell from the particle's initial location, choosing the shortest time and checking this time against t_1. This is shown in Figure B2 for two particles following two streamlines ($A_{0,1}$ and $B_{0,1}$) each exiting the cell at different faces. Equation (B9) may be re-written as $\Delta t = \frac{1}{b}\ln\left(\frac{a_1}{v_{px}}\right)$, where a_1 is the velocity at the exit face of interest.
4. If t_1 is much longer than the time the particle needs to travel through one cell, it will be passed from one cell face to the next, following interpolated streamlines, in a single step per cell. This results in tremendous computational savings, regardless of the magnitudes of the velocities in the domain, and makes calculations with very large numbers of particles (50,000-100,000) practical. This method is similar to "temporal sub-cycling" methods proposed for Eulerian-based advection schemes.

5. Once the particle has reached time t_1, all remaining particles are advected to their corresponding locations at time t_1, one by one.

Dispersive step

Further complexities are added to the model when features of flow other than purely advective transport are considered. Chemical interactions that affect only the cell velocity or the mass of a given particle may be included in the above method with little modification (see below). Interactions that are strongly a function of the local constituent concentration at small time scales are more difficult to implement, and are not considered in this version of the model. Wells are handled explicitly, assuming radial flow in any grid block that contains a well screen. Particles that are about to enter a block containing a well are removed from the domain. The cell face velocities are used to estimate the time the particle would reach the well, and that particle's mass is added to that well's mass capture profile at that time.

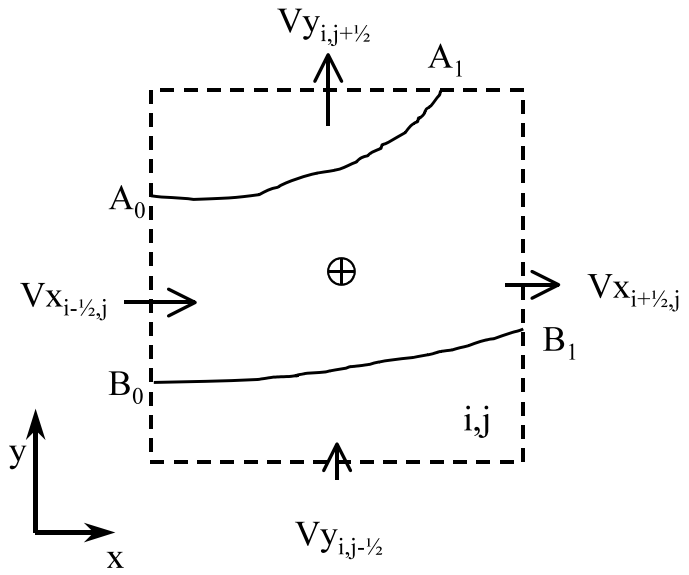


Figure B2. Cartoon of two particle streamlines in a cell.

Local dispersive phenomena, that is velocity gradients that increase contaminant mixing on a length scale less than those explicitly considered by the heterogeneity introduced into the domain, can be difficult to implement computationally. Accuracy considerations of various velocity interpolation schemes and their impact on solutions to (A2) have been investigated by LaBolle, et al. (1996). LaBolle and his co-authors suggest BiLinear velocity interpolation as the most accurate method of solving for the advection-correction and random-walk dispersion terms in Equation (B3). BiLinear velocity interpolation has been extended to three dimensions as described below and is used for the local dispersion terms in SLIM-FAST. The advective terms are still handled as detailed above. For the velocity and particle locations as defined in Figure B3 (for three-dimensions centered on *(i, j, k),* the velocity in the *x*-direction, as interpolated BiLinearly, may be written as:

$$v_{xBLI} = \sum_{n=1}^{8} \theta_n v_{xn} \tag{B12}$$

where the BiLinear interpolation coefficients are:

$$\begin{aligned}
\theta_1 &= (1-F_x)(1-F_y)(1-F_z) \\
\theta_2 &= F_x(1-F_y)(1-F_z) \\
\theta_3 &= F_x F_y(1-F_z) \\
\theta_4 &= (1-F_x)F_y(1-F_z) \\
\theta_5 &= (1-F_x)(1-F_y)F_z \\
\theta_6 &= (1-F_x)(1-F_y)F_z \\
\theta_7 &= (1-F_x)(1-F_y)F_z \\
\theta_8 &= (1-F_x)F_y F_z
\end{aligned} \tag{B13}$$

and the corresponding surrounding velocities are

$$\begin{aligned}
v_{x1} &= \frac{v_x\left(i-\frac{1}{2}, j-\frac{1}{2}, k-\frac{1}{2}\right)+ v_x\left(i+\frac{1}{2}, j-\frac{1}{2}, k-\frac{1}{2}\right)}{2} \\
v_{x2} &= \frac{v_x\left(i+\frac{1}{2}, j-\frac{1}{2}, k-\frac{1}{2}\right)+ v_x\left(i+\frac{3}{2}, j-\frac{1}{2}, k-\frac{1}{2}\right)}{2} \\
v_{x3} &= \frac{v_x\left(i+\frac{1}{2}, j+\frac{1}{2}, k-\frac{1}{2}\right)+ v_x\left(i+\frac{3}{2}, j+\frac{1}{2}, k-\frac{1}{2}\right)}{2} \\
v_{x4} &= \frac{v_x\left(i-\frac{1}{2}, j+\frac{1}{2}, k-\frac{1}{2}\right)+ v_x\left(i+\frac{1}{2}, j+\frac{1}{2}, k-\frac{1}{2}\right)}{2} \\
v_{x5} &= \frac{v_x\left(i-\frac{1}{2}, j-\frac{1}{2}, k-\frac{1}{2}\right)+ v_x\left(i+\frac{1}{2}, j-\frac{1}{2}, k+\frac{1}{2}\right)}{2} \\
v_{x6} &= \frac{v_x\left(i+\frac{1}{2}, j-\frac{1}{2}, k+\frac{1}{2}\right)+ v_x\left(i+\frac{3}{2}, j-\frac{1}{2}, k+\frac{1}{2}\right)}{2} \\
v_{x7} &= \frac{v_x\left(i+\frac{1}{2}, j+\frac{1}{2}, k+\frac{1}{2}\right)+ v_x\left(i+\frac{3}{2}, j+\frac{1}{2}, k+\frac{1}{2}\right)}{2} \\
v_{x8} &= \frac{v_x\left(i-\frac{1}{2}, j+\frac{1}{2}, k+\frac{1}{2}\right)+ v_x\left(i+\frac{3}{2}, j+\frac{1}{2}, k+\frac{1}{2}\right)}{2}.
\end{aligned} \tag{B14}$$

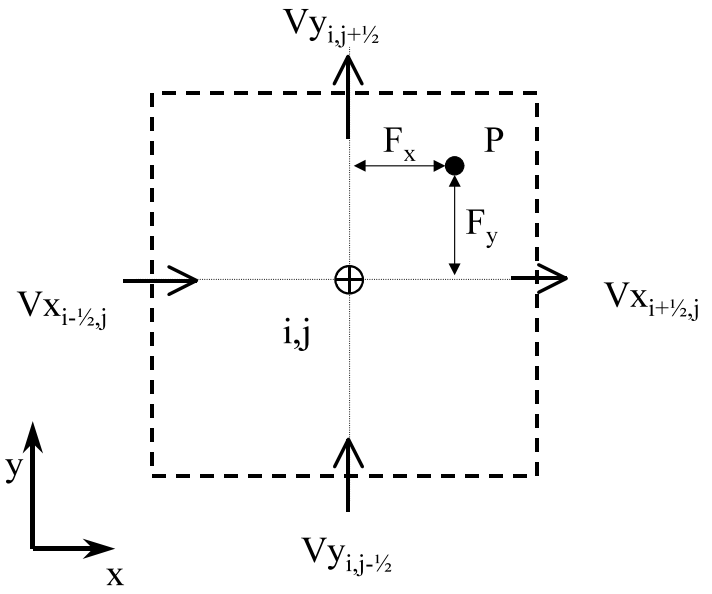


Figure B3. Figure depicting coefficients used in BiLinear interpolation.

Retardation effect

As defined in Appendix A, the retardation factor R_j represents the instantaneous state of partitioning between the aqueous and immobile fractions of species *j* at a point in space and time. In the current particle application, the magnitude of R_j for each sorbing radionuclide (j) was evaluated as a function of the ambient CHESHIRE groundwater composition, which was assumed to be fixed throughout the domain, and the local mineralogy, which was assumed to be spatially variable (Appendix H). For evaluation purposes the definition of R_j (Equation D1) was decomposed into the following form

$$R_j(\mathbf{x},t) = 1 + \frac{u_j^{im}}{u_j} = 1 + \sum_i \widehat{R}_j^i(\mathbf{x},t) \tag{B15}$$

where the sum represents the individual (and independent) contributions of different sorbing minerals (i) to the overall immobile mole fraction of radionuclide j at location **x** and time t. This factor may be "applied" through modification of advective and dispersive displacements as reviewed above

B.5 Radioactive Decay Processes

The effects of radioactive species decay can be accounted for in two ways. Simple decay of a radionuclide into a daughter product may be accounted for at any time in a postprocessing step by modifying the calculated, undecayed concentration via

$$c_j^{Td}(\mathbf{x},t) = c_j^T(\mathbf{x},t) \cdot e^{-\lambda_j t}, \tag{B16}$$

where λ_j is the decay rate for species j. This method will only yield valid results if accumulations of the daughter product are of no interest in the problem and if the loss of the radionuclide over the course of a simulation does not affect its mobility in the system.

Alternatively, real-time decay of a radionuclide may be implemented within a particle simulation by incrementally removing its mass and transferring it to the inventory of its daughter product. This approach must be pursued if daughter product ingrowth is important over the course of the simulation, or if the loss of the radionuclide affects its mobility or retardation in the system.

For example, this issue will be of concern for three radionuclides that are members of the ^{241}Pu decay chain, e.g.,

$$^{241}\mathrm{Pu} \xrightarrow{\lambda_{241\mathrm{Pu}}} {}^{241}\mathrm{Am} \xrightarrow{\lambda_{241\mathrm{Am}}} {}^{237}\mathrm{Np} \xrightarrow{\lambda_{237\mathrm{Np}}} \ldots \tag{B17}$$

This issue arises because the half-lives of ^{241}Pu and ^{241}Am (14.4 y and 43.3 y, respectively) may be small enough to render the ingrowth of ^{241}Am and ^{237}Np significant over the course of a practical simulation. Because the half-life of ^{237}Np is large (2.14×10^6 y) , its daughter products will not accrue appreciably over the course of a practical simulation, and so its decay will only be considered as part of a postprocessing step.

Suppose we have $N_{241\mathrm{Pu}}$ moles of ^{241}Pu at some location $\mathbf{x}$ and time t. Over an increment of time Δt, a portion of this mass will decay and produce some amount of ^{241}Am, ^{237}Np, and other daughter products of ^{237}Np. If we assume the decay rate of ^{237}Np is approximately zero, then the loss of ^{241}Pu over this time is

$$\Delta N_{241\mathrm{Pu}} = -N_{241\mathrm{Pu}}(\mathbf{x},t)\cdot\left(1-e^{-\lambda_{241\mathrm{Pu}}\Delta t}\right) \tag{B18a}$$

and leads to the production of the following amounts of ^{241}Am and ^{237}Np,

$$\Delta N_{241\mathrm{Am}} = N_{241\mathrm{Pu}}(\mathbf{x},t)\cdot\frac{\lambda_{241\mathrm{Pu}}}{\lambda_{241\mathrm{Am}}-\lambda_{241\mathrm{Pu}}}\left(e^{-\lambda_{241\mathrm{Pu}}\Delta t}-e^{-\lambda_{241\mathrm{Am}}\Delta t}\right) \tag{B18b}$$

and

$$\Delta N_{237\mathrm{Np}} = N_{241\mathrm{Pu}}(\mathbf{x},t)\cdot\left(1-\frac{\lambda_{241\mathrm{Am}}}{\lambda_{241\mathrm{Am}}-\lambda_{241\mathrm{Pu}}}e^{-\lambda_{241\mathrm{Pu}}\Delta t}-\frac{\lambda_{241\mathrm{Pu}}}{\lambda_{241\mathrm{Am}}-\lambda_{241\mathrm{Pu}}}e^{-\lambda_{241\mathrm{Am}}\Delta t}\right). \tag{B18c}$$

This suggests that a transition probability can be identified for simulating radionuclide decay in the particle model. Over an increment of time Δt, the probabilities that a particle of ^{241}Pu will decay into a particle of ^{241}Am or ^{237}Np are just

$$P_{Pu\to Am} = \frac{\Delta N_{241\mathrm{Am}}}{N_{241\mathrm{Pu}}(\mathbf{x},t)} = \frac{\lambda_{241\mathrm{Pu}}}{\lambda_{241\mathrm{Am}}-\lambda_{241\mathrm{Pu}}}\left(e^{-\lambda_{241\mathrm{Pu}}\Delta t}-e^{-\lambda_{241\mathrm{Am}}\Delta t}\right) \tag{B19a}$$

and

$$P_{Pu\to Np} = \frac{\Delta N_{237\mathrm{Np}}}{N_{241\mathrm{Pu}}(\mathbf{x},t)} = 1 - \frac{\lambda_{241\mathrm{Am}}}{\lambda_{241\mathrm{Am}} - \lambda_{241\mathrm{Pu}}} e^{-\lambda_{241\mathrm{Pu}}\Delta t} - \frac{\lambda_{241\mathrm{Pu}}}{\lambda_{241\mathrm{Am}} - \lambda_{241\mathrm{Pu}}} e^{-\lambda_{241\mathrm{Am}}\Delta t}. \quad \text{(B19b)}$$

Similarly, suppose we independently consider N_{241}Am moles of ^{241}Am at the same location **x** and time *t*, different from any produced by ^{241}Pu decay. Over the same increment of time, a portion of this mass will also decay and produce some amount of ^{237}Np, and other daughter products of ^{237}Np, independent from amounts associated with ^{241}Pu decay. Since the decay rate of ^{237}Np is approximately zero, then the loss of the initial ^{241}Am over this time will produce

$$\Delta N_{237\mathrm{Np}} = N_{241\mathrm{Am}}(\mathbf{x},t) \cdot \left(1 - e^{-\lambda_{241\mathrm{Am}}\Delta t}\right) \quad \text{(B20)}$$

such that the probability of an ^{241}Am particle decaying to ^{237}Np over Δt is

$$P_{Am\to Np} = \frac{\Delta N_{237\mathrm{Np}}}{N_{241\mathrm{Am}}(\mathbf{x},t)} = 1 - e^{-\lambda_{241\mathrm{Am}}\Delta t} \quad \text{(B21)}$$

If the time step is sufficiently small, these probabilities can be approximated by

$$P_{Pu\to Am} \sim \lambda_{241\mathrm{Pu}}\Delta t \quad \text{(B22a)}$$

$$P_{Pu\to Np} \sim \frac{\lambda_{241\mathrm{Pu}}\lambda_{241\mathrm{Am}}\Delta t^2}{2}, \quad \text{(B22b)}$$

or

$$P_{Am\to Np} \sim \lambda_{241\mathrm{Am}}\Delta t \quad \text{(B22c)}$$

As an example, the transitional probability decay results of equations (B19) and (D21) were applied to 100,000 particles of ^{241}Pu located within an elemental volume of a static, nonflowing system. In a series of 1000 one-year time steps, particles of ^{241}Pu were randomly transferred to ^{241}Am or ^{237}Np particles according to equation (B19). As particles of ^{241}Am accrued in this process, they were also randomly transferred to ^{237}Np particles according to equation (D21). As shown in Figure B4, the net result is consistent with the ingrowth predicted from the analytic Bateman equations (e.g., Friedlander et al., 1981).

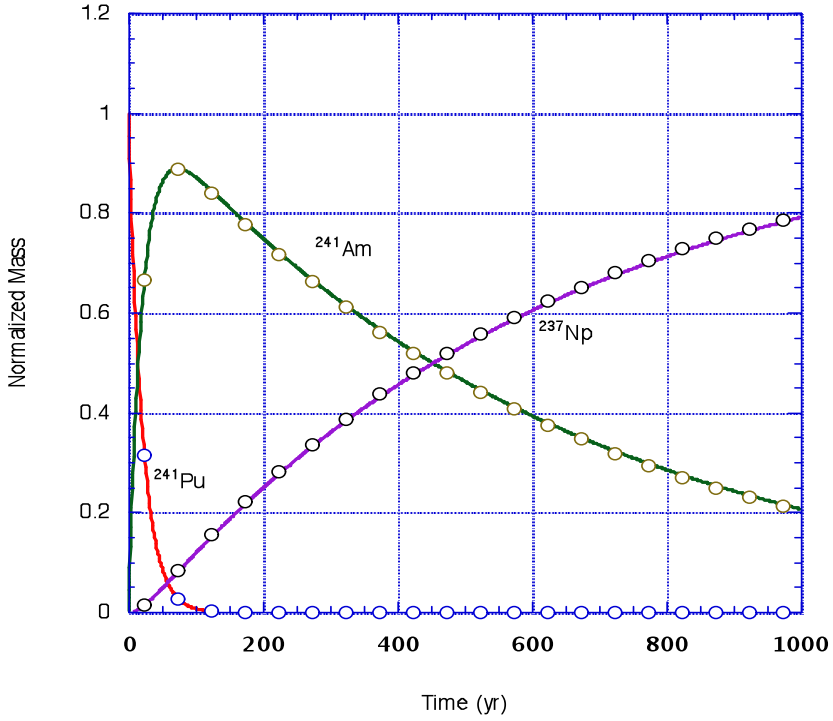


Figure B.4. Normalized ingrowth of ^{241}Am and ^{37}Np from the decay of an initial unit mass of ^{241}Pu over 1000 yr. Solid curves represent the Bateman solution; symbols represent a computed particle solution based on 100,000 particles and use of the transitional probability model (D10) with 1 yr steps.

B.5 Matrix Diffusion Processes

The effects of matrix diffusion can be implemented into the particle model using a similar transitional probability approach, as suggested by the approximate model of Liu et al. (2000). This approach is based upon a simplified conceptual model of the fractured system in which typical parallel fractures are of a width 2*b* and are separated from one another by a typical distance 2*S*. During each time step, particles in the fractured regime may be transferred into stationary particles in the matrix regime, and vice versa, as a probabilistic function of the time step, various geometrical factors associated with the idealized fracture system model, and several physicochemical properties of the matrix blocks. In our case, we assume that these exchange processes occur only because of diffusion into and out of the matrix blocks. Particles located in the matrix are not advected or diffused according to a model like (B3); rather, their position remains fixed until they are probabilistically returned to the fracture regime.[2]

For radionuclide species *j*, the probability of moving from the fractures into the matrix is given by

[2] Nevertheless, the exchange probabilities do account for the diffusive processes that occur in the matrix, as they are functions of the block size, *S*.

$$P_{j,fm} \sim \frac{3D_{e,m}}{Sb R_{j,m}} \Delta t \qquad \text{(B23a)}$$

where $D_{e,m}$ and $R_{j,m}$ are, respectively, the effective molecular diffusivity and retardation coefficient for radionuclide *j* in the matrix. Conversely, the probability of moving from the matrix into the fractures is given by

$$P_{j,mf} \sim \frac{3D_{e,m}}{S^2 R_{j,m}^2 \phi_m} \Delta t \qquad \text{(B23b)}$$

where ϕ_m is the matrix porosity. Our experimental tests indicate that the provisional time step associated with these transitional terms should be chosen such that $P_{j,fm}$ and $P_{j,mf}$ are less than 0.02.

Figure B5 shows results of a simple test of this algorithm based upon a one-dimensional test problem. Here conservative tracer migration was induced in a 200 m fractured column with a seepage velocity V of 100 m/yr. A 10^{-10} mol/L solution of tritium ($R_{tritium,m} \sim 1$) was injected at the upstream boundary of the fractured regime for a period of 100 yr. The injected pulse employed 25,000 particles. The fracture properties are $b \sim 0.005$ m, $S \sim 0.495$ m, $\phi_m \sim 0.15$, and $D_{e,m} \sim 3.15 \times 10^{-3}$ m^2/yr (10^{-6} cm^2/s).

The results in Figure B5 show the particle breakthrough as influenced by implementation of (B22), and as compared with a two-dimensional discrete fracture simulation of the same problem using the GIMRT reactive transport model[3] (Steefel and Yabusaki, 1996; Pawloski et al., 2001).

B.6 SLIM-FAST Memory Requirements

Table B1 shows a calculation of the approximate memory requirements for a SLIM-FAST problem using the major arrays in the code. The calculation is based on a 1 million node problem (100 x 100 x 100 cells) with 1 million particles. It is for a case where SLIM-FAST calculates velocities internally and is show for one or two constituents. This will be peak memory usage, as arrays are allocated and de-allocated as needed within the code.

[3] Here the second spatial coordinate axis points into the matrix block

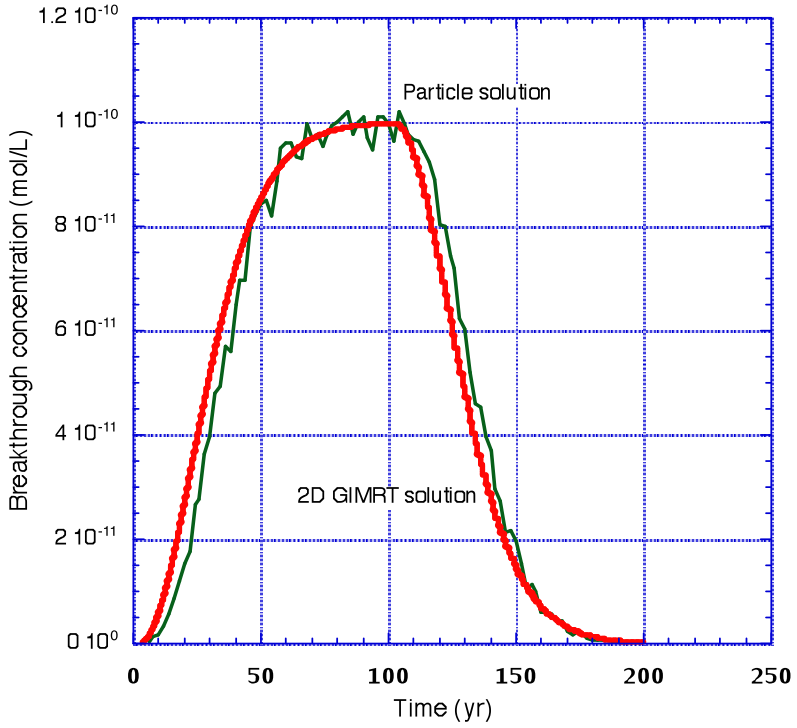


Figure B5. Particle breakthrough profile for tritium (thin line) reflecting the influence of matrix diffusion, as implemented with the transitional probability equations (D11) in a 1-D test problem. The thicker line represents the solution provided by a 2-D, discrete fracture application of the GIMRT model to the same problem.

Table B1. Approximate memory requirements for a SLIM-FAST problem based on a 1 million node grid, 1 or 2 species, and 1 million particles

Variable	Memory Usage (KB)	
	1 constituent	2 constituents
nx	100	
ny	100	
nz	100	
npmax	1,000,000	
kx	7,813	7,813
ky	7,813	7,813
kz	7,813	7,813
head	7,813	7,813
phi	7,813	7,813
R	7,813	15,625
katt	7,813	15,625
kdet	7,813	15,625
conc	3,906	7,813
part	156,250	156,250
ic_cat	3,906	3,906
V	23,438	23,438
ipwell	78,125	78,125
slim total	328,125	355,469

Appendix C: SLIM-FAST Limitations and Wish List

1. Variable grid. The current version of SLIM-FAST uses a regular grid, i.e., dx is constant, etc. While an ad hoc version has been coded to do some forms of variable grid simulations (dx = dx(x), etc), this has not been formalized across all features of SLIM-FAST, such as the variety of IC options, wells, flux planes, etc.
2. Boundary condition upgrade. SLIM-FAST currently does not do no-flow boundary conditions with particle reflections, only constant flux BC's. Adding more BC options in the input and adding reflections in the code is desirable.
3. More IO options. It would be nice to read/write other file types. VTK support has been added which allows SLIM-FAST concentrations to be read by VisIt, but SILO I/O would be helpful.
4. Parallel implementation. The current version of SLIM-FAST is serial. While particle methods do not parallelize well on distributed memory machines, an option to split the main particle loop on shared-memory machines with multiple processors is on the wish list.