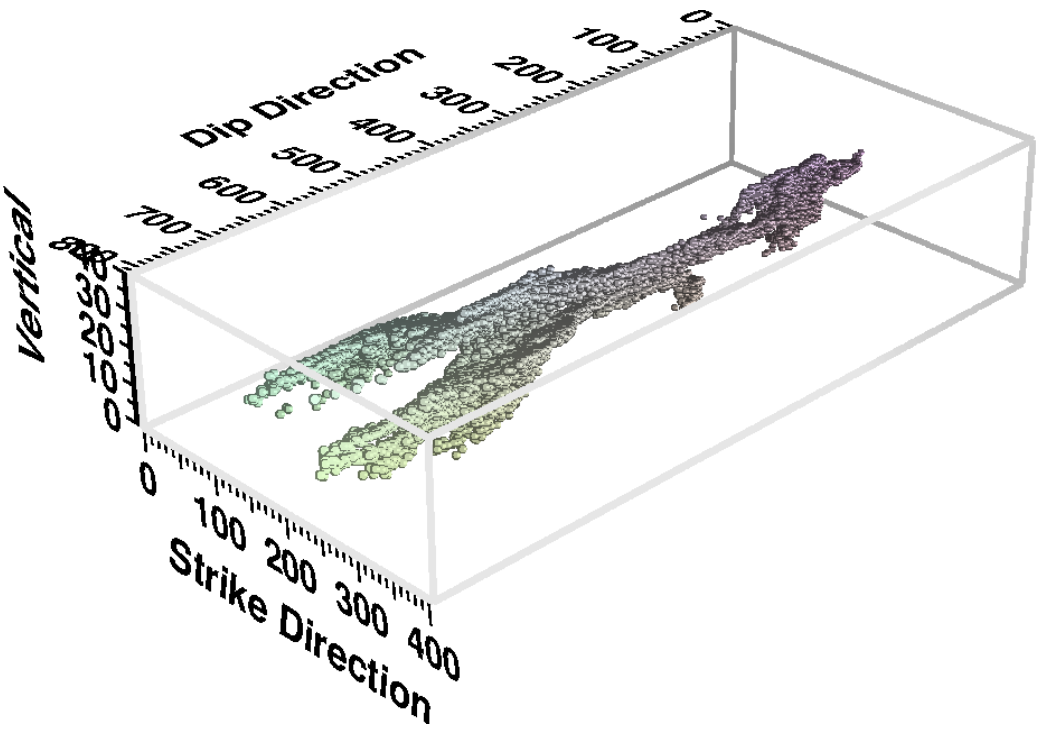
RWHet: **R**andom **W**alk Particle Model for Simulating Transport in **Het**erogeneous Permeable Media

Version 3.2

User's Manual and Program Documentation

Eric M. LaBolle
11/2006



Disclaimer

RWHet was developed for the benefit of hydrologic sciences and scientists. RWHet is officially unsupported by the author, although he welcomes questions and will generally seek to answer them provided that you've read this document and he has the time. As with any scientific software, the user should thoroughly understand its applicability, limitations and foundational theory before using it. Use of RWHet is at your own risk; the author cannot guarantee the accuracy of RWHet for your application. The author assumes no liability for the consequences of your use of RWHet. RWHet is the sole ownership of the author and RWHet or any of the code therein may not be distributed without this document and the author's permission.

Recent Changes

5-9-08: At the request of users, particle (prt) file output is now unformatted binary to avoid large ASCII prt files. A prt file unformatted binary to ASCII conversion program PRTREADER is provided.

1-1-09: The time step control has been changed so that dtcntrl now scales the maximum dispersive displacement. A value of dtcntrl = 1.0 yields a maximum displacement of a ½ cell.

Table of Contents

1.0 Overview

The program RWHet solves an advection-dispersion equation (ADE) on a block-centered finite-difference flow solution (e.g., as provided by MODFLOW [*McDonald, M.G. and Harbaugh*, 1988]) by the random-walk particle method (RWPM) [*LaBolle et al.*, 1996; 1998; 2000]. RWHet is written in FORTRAN90 and command line driven. The current stand-alone version is limited to a rectilinear grid with constant discretization in the domain of interest. This document describes the mathematical concepts behind RWHet, the program design, and how to use it.

2.0 The Random Walk Particle Method (RWPM)

RWHet solves an advection dispersion equation of the form

$$\begin{aligned}&\frac{\partial}{\partial t}\left[R\Theta(\mathbf{x},t)c(\mathbf{x},t)\right]= \\ &-\sum_i \frac{\partial}{\partial x_i}\left[v_i(\mathbf{x},t)\Theta(\mathbf{x},t)c(\mathbf{x},t)\right]+\sum_{i,j}\frac{\partial}{\partial x_i}\left[\Theta(\mathbf{x},t)D_{ij}(\mathbf{x},t)\frac{\partial c(\mathbf{x},t)}{\partial x_j}\right] \\ &+\sum_k q_k(\mathbf{x},t)c_k(\mathbf{x},t)\delta_k(\mathbf{x}-\mathbf{x}_k)-\lambda'\left[R\Theta(\mathbf{x},t)c(\mathbf{x},t)\right]\end{aligned} \tag{2.1}$$

where c [ML^{-3}] is concentration, $\mathbf{v}$ [LT^{-1}] is a velocity, Θ [L^3L^{-3}] is effective volumetric water content (or porosity), $R = 1 + K_d\rho_b/\Theta$ is a dimensionless retardation factor, K_d [L^3M^{-1}] is a soil distribution coefficient, the ratio of the solid phase [MM^{-1}] to aqueous phase [ML^{-3}] concentrations at equilibrium, ρ_b is the soil bulk density [ML^{-3}], λ' is a decay coefficient [T^{-1}], c_k [ML^{-3}] is the aqueous phase concentration in the flux q_k [L^3T^{-1}] of water at $\mathbf{x}_k$, $\mathbf{D}$ [L^2T^{-1}] is a real symmetric dispersion tensor given as

$$D_{ij} = \left(|\mathbf{v}|\alpha_T + D^*\right)\delta_{ij} + \left(\alpha_T - \alpha_L\right)v_i v_j / |\mathbf{v}| \tag{2.2}$$

where α_T and α_L [L] are transverse and longitudinal dispersivities, respectively, and D^* is effective molecular diffusivity [L^2T^{-1}]. Let $\tilde{C} = \Theta R c$, where $\tilde{C}$ is the volumetric concentration in the subsurface (mass per unit volume). Then one can write (2.1) as

$$\frac{\partial \tilde{C}}{\partial t} = -\sum_i \frac{\partial}{\partial x_i}\left[v_i \tilde{C}/R\right] + \sum_{i,j} \frac{\partial}{\partial x_i}\left[\Theta D_{ij} \frac{\partial}{\partial x_j}\left(\frac{\tilde{C}}{\Theta R}\right)\right] + \sum_k q_k c_k \delta_k\left(\mathbf{x}-\mathbf{x}_k\right) - \lambda' \tilde{C} \tag{2.3}$$

Applying the chain rule to the dispersion term and rearranging yields:

$$\frac{\partial \tilde{C}}{\partial t} = -\sum_i \frac{\partial}{\partial x_i}\left[\frac{\tilde{C}}{R}\left(v_i + \sum_j \frac{\partial D_{ij}}{\partial x_j} + \sum_j \frac{D_{ij}}{\Theta}\frac{\partial \Theta}{\partial x_j}\right)\right] + \sum_{i,j} \frac{\partial^2}{\partial x_i \partial x_j}\left(\frac{D_{ij}\tilde{C}}{R}\right) + \sum_k q_k c_k \delta_k\left(\mathbf{x}-\mathbf{x}_k\right) - \lambda' \tilde{C} \tag{2.4}$$

This equation is in the form of a Fokker-Planck equation. Standard random-walk methods (e.g., *Kinzelbach* [1988] and *Tompson et al*. [1987; 1990; 1992; 1993]) approximate solutions to equation (2.1) by simulating sample (particle) paths corresponding to diffusion processes described by stochastic differential equations (SDEs). Here, "diffusion processes" refers to Markov processes with continuous sample paths as mathematical models of real subsurface-transport phenomena. Therefore, the RWPM simulates a "diffusion process," which in the present context is an "advection dispersion process."

In the RWPM, the concentration of an aqueous solute is represented by a finite system of N_p particles of mass $m_p(t)$ via [*LaBolle et al.*, 1996]

$$R\Theta(\mathbf{x})c^p(\mathbf{x},t) = \sum_{p\in N_p} m_p(t)\,\delta\left(\mathbf{x}-\mathbf{X}_p(t)\right) \tag{2.5}$$

where δ is a Dirac function $\mathbf{X}_p(t)$ is the location of particle p at time t and c is the concentration. Because (2.5) is discontinuous, a modified expression of the form

$$R\Theta(\mathbf{x})c^s(\mathbf{x},t) = \sum_{p\in N_p} m_p(t)\,\zeta\left(\mathbf{x}-\mathbf{X}_p(t)\right) \tag{2.6}$$

is typically used to "smooth" the spatial distribution of concentration, where ζ is an interpolation, or projection function [*Bagtzoglou et al.*, 1992] normalized such that $\int_\Omega \zeta dV = 1$, where Ω is the domain of porous medium. The degree of smoothness obtained is

controlled by the shape of ζ and the particle resolution, N_r, the number of particles used to represent an arbitrary unit of mass. The total mass M in Ω is given as $M = \int_\Omega R\Theta c dV = \sum_{p\in Np} m_p$.

2.1 D as a Smooth Function of Space

Where D_{ij} and Θ are interpreted as smooth functions of space, simulation of advective and diffusive mass transport may proceed by changing particle positions with time via an *Itô-Euler* integration scheme given as [*Gardiner*, 1990]

$$X_i(t+\Delta t) - X_i(t) = A_i(\mathbf{X},t)\Delta t + B_{ij}(\mathbf{X},t)\Delta W_j(\Delta t) \tag{2.7}$$

where A_i is a "drift" vector [LT^{-1}], B_{ij} is a tensor [$LT^{-1/2}$] defining the strength of diffusion, and W_j, a Wiener process [$T^{1/2}$], is a vector of independent normally-distributed random variables with zero mean and covariance

$$\left\langle \Delta W_i \Delta W_j \right\rangle = \delta_{ij}\Delta t \tag{2.8}$$

The particle mass density $f \equiv R\Theta c^p$ obtained through repeated use of (2.3) on all particles satisfies the *Itô Fokker-Planck* Equation [*Risken*, 1989]

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial x_i}\left(A_i f\right) - \frac{1}{2}\frac{\partial^2}{\partial x_i \partial x_j}\left(B_{ik}B_{kj} f\right) = 0 \tag{2.9}$$

in the limit as $N_r \to \infty$ and $\Delta t \to 0$, where

$$A_i = \lim_{\Delta t\to 0}\frac{1}{\Delta t}\left\langle X_i(t+\Delta t) - X_i(t)\right\rangle \tag{2.10}$$

$$B_{ik}B_{kj} = \lim_{\Delta t\to 0}\frac{1}{\Delta t}\left\langle \left[X_i(t+\Delta t) - X_i(t)\right]\left[X_j(t+\Delta t) - X_j(t)\right]\right\rangle \tag{2.11}$$

To simulate transport according to (2.1), we specify A_i and B_{ij} in (2.9) as [*Kinzelbach*, 1988; *Uffink*, 1988; *Tompson et al.*, 1987]

$$A_i = \frac{1}{R}\left[v_i + \frac{\partial D_{ij}}{\partial x_j} + D_{ij}\Theta^{-1}\frac{\partial \Theta}{\partial x_j}\right] \tag{2.12}$$

$$\frac{2D_{ij}}{R} = B_{ik}B_{kj}, \tag{2.13}$$

Substituting (2.12) into (2.7) yields the equation for a particle displacement

$$X_i(t+\Delta t) - X_i(t) = \frac{1}{R}\left(v_i + \frac{\partial D_{ij}}{\partial x_j} + D_{ij}\Theta^{-1}\frac{\partial \Theta}{\partial x_j} \right)\Delta t + B_{ij}(\mathbf{X},t)\Delta W_j(\Delta t) \quad (2.14)$$

where coefficients in the right-hand-side of (2.14) are evaluated at $X_i(t)$. Because D_{ij} is real and symmetric, elements of the tensor B_{ij} can be calculated by diagonalizing D_{ij}**,** taking the positive root of the eigenvalues, transforming back and multiplying by an arbitrary orthogonal matrix, [*Risken*, 1989] (for details refer to *Tompson et al.* [1987]). This development has been extended to treat reactive transport for constituents that undergo binary reversible equilibrium sorption [*Tompson*, 1993] and kinetic decay [*Tompson and Dougherty*, 1992].

2.2 D as a Discrete Function of Space

The case where D_{ij} is a discrete function of space (i.e., a step function along some internal boundary) will only be considered for isotropic $\underline{D_{ij}}$. In this case, simulation of advective and diffusive mass transport may proceed by changing particle positions with time via (*LaBolle and Zhang*, 2006)

$$X_i(t+\Delta t) - X_i(t) = \frac{1}{R}\left(v_i(\mathbf{X},t) + \Theta^{-1} D_{ij}\frac{\partial \Theta}{\partial x_j} \right)\Delta t + \sum_j \hat{B}_{ij}\left[X_k + \hat{B}_{lk}(\mathbf{X},t)\Delta W_k, t \right]\Delta W_j(\Delta t) \quad (2.15a)$$

$$\hat{B}_{ij}(\mathbf{x},t) = \sqrt{\lambda(\mathbf{x},t)}\,\delta_{ij} \quad (2.15b)$$

where λ is the eigenvalue of $2\mathbf{D}/R$**,** and in the second term on the rhs of (2.15a), $\hat{B}_{ij}$ is evaluated at time t and location given by the vector whose lth component is $X_l + \hat{B}_{lk}(\mathbf{X},t)\Delta W_k$.

3.0 Input and Output

The state of the system is given by particle locations and masses. The following sections describe details of the numerical implementation. RWHet operates from a command line requesting the main input and output file, as well as an additional output file that logs particle information if and when they enter and exit the domain. The main output file

contains a brief summary of the input and information generated during a simulation. A detailed description of the required input and output files is given below.

3.1 Main Input File

The user can comment the main input file (previous page) by specifying the pound sign # as the first character of a line containing a comment. In the description that follows, line numbers do not include comments and correspond with those shown in the example.

MAIN INPUT FILE

```
     # run titles
1.   LLNL Simulation: Pump-and Treat Scenario
2.   1-99
     # debug output (0 none) (1 – 3 increasing output)
3.   0
     # debug file
4.   r1.dbg
     # nx, ny, nz
5.   81 81 81
     # dx, dy, dz
6.   5.0 10.0 0.5
     # extent of model domain:
     # ixmin,ixmax
     # iymin,iymax
     # izmin,izmax
7.   1 81
8.   1 81
9.   1 81
     # reflect (1) or absorb (0) BCs at ixmin, ixmax
10.  0 0
     # reflect (1) or absorb (0) BCs at iymin, iymax
11.  0 0
     # reflect (1) or absorb (0) BCs at izmin, izmax
12.  0 0
     # maximum number of particles
13.  100000
     # random number seeds
14.  30003 30005
     # interp, bke (1.0 forward, -1.0 backward)
15.  0 1.0
     # cres, npres
16.  0.01  1.0
     # maxt, dtinit, initt, tcntrl
17.  36500.0 365.000 0.0 3
     # output control (opc)
18.  r1.opc
     # parameters (par)
19.  r1.par
     # velocity input (vlc)
20.  r1.vlc
     # internal boundary conditions (bnd)
21.  r1.bnd
     # point particle input (pnt)
22.  r1.pnt
     # sampling locations (sam)
23.  r1.sam
```

1 – 2. (CHARCTER*80) Run titles.

3. (INTEGER *idbg*) Specifies the level of debug output which can often be useful in identifying problems with model input. A “0” is used to specify no debug output.

4. (CHARCTER*80 *dbgfl*) The debug file name up to 80 characters.

5. (INTEGER *nx, ny, nz*) The model domain in terms of the number of cells *nx*, *ny*, *nz* in the *x*, *y* an *z* directions. All must be greater than 0. Particle displacements do not occur in directions corresponding to dimensions of “1.” This allows the user to perform 1, 2 and 3-D simulations.

6. (REAL *dx, dy, dz*) The constant discretization of the model grid: Δx, Δy , and Δz. All must be positive.

7 – 8. (INTEGER *ixmin, ixmax …*) Cell numbers corresponding to the extent of the active model in the *x* (line 7), *y* (line 8), and *z* (line 9) directions. The active domain includes these cell numbers and can be smaller than *nx, ny, nz*, but not larger.

10 – 12. (INTEGER *ixm, ixp …*) Reflecting (1) or absorbing (0) boundary conditions [*LaBolle et al.,* 1996] at extents of the active model domain in the *x* (line 10), *y* (line 11) and *z* (line 12) directions. The first number corresponds to the minimum extent of the boundary of the active domain, the second number to the maximum.

13. (INTEGER *npmax*) Maximum number of particles. If exceeded during the simulation, simulation will terminate.

14. (INTEGER *iseed1 iseed2*) Random number seeds used in random number generators. Use large odd integers on the order of 30,001.

15. (INTEGER *interp*, REAL *bke*) (a) Specifies use of the bilinear interpolation [*LaBolle et al.,* 1996], or block discrete **D**. (b) Specifies use of the forward in time ADE (1.0) or backward in time ADE (-1.0) (i.e., the velocity field is reversed, see *Fogg et al.*, [1998]).

16. (REAL *cres*, INTEGER *nres*) (a) Concentration corresponding to the particle resolution. (b) Particle resolution as described in section 2.0. If *nres* = 1, no particle splitting is performed. If *nres* > 1, the concentration field is resolved by splitting particles to ensure at least *nres* particles in a cell with concentration $c \geq cres$.

17. (REAL *maxt*, REAL *dtinit*, REAL *initt*, REAL *tcntrl*) (a) Time to end simulation. (b) Maximum time step. (c) Initial time for start of simulation. (d) Time step control, generally taken as < 1.0.

18 – 23. (CHARACTER*80) Other input files listed in required order in the example and described in detail later.

3.2 Output Control File

The output control file specifies model output file names as well as corresponding times for output to each file.

```
OUTPUT CONTROL FILE
1.   r1.mom
2.   r1.con
3.   r1.mdt
4.   r1.bth
5.   r1.prt
6.   r1.int
7.   r1.mon
8.    00.0 1 2 0 1 0 1 1
9.    36.5 1 0 0 0 1 1 1
10.   73.0 1 0 0 0 0 1 1
        .  . . . . . .
        .  . . . . . .
```

1 – 6 (CHARACTER*80) Output file names corresponding to output containing (1) spatial moments (*.mom), (2) concentrations (*.con), (3) unused at present, (4) breakthrough at extents of the active domain (*.bth), (5) particle attributes (*.prt) , (6) internal boundary condition counters (*.int), and (7) concentration monitoring (*.mon).

7 … (REAL *tnextopc*, INTEGERS opc(1), opc(2), …) Times for output to be generated followed by integers specifying the type of output to be generated. The integer fields refer in order from left to right to the output files specified. A “0” means that output will not be generated; a “1” will generate output at the specified time. For concentration (*.con) and internal boundary condition counter (*.int) output, a “1” will generate information for particle numbers, and a “2” will generate information on concentration and particle mass, respectively. In the example, output is generated at time 36.5 for spatial moments, internal boundary condition counters, and concentration monitoring.

3.3 Parameter Fields Input

Parameter fields include effective volumetric water content or porosity, longitudinal and transverse dispersivity, diffusion coefficient, and retardation factor. Currently, RWHet allows for the specification of these fields as either constants, zones determined from indicators stored in an ASCII or binary grid file compatible with the geostatistical software TPROGS TSIM [*Carle*, 1998; *Carle et al.*, 1998], or as parameters fields.

```
PARMETER FIELD INPUT
(FIELDS AND CONSTANTS)
1.   0
2.   POROSITY
3.   1  “porfl.txt”
4.   RETARD
5.   0   1.0
6.   LONGITUDINAL DISPERSIVITY
7.   0   0.01
8.   TRANSVE0SE DISPERSIVITY
9.   0   0.01
10.  DIFFUSION COEFFICIENT
11.  -1  “difffl.bin”
12.  DECAY
13.  0   0.000
```

1. (INTEGER *nzone*) Number of zones

in the TPROGS TSIM BGR or ASC file. If *nzone* = 0, The number of zones is equal to the number of nodes, and parameters may be specified as either constants for all zones or parameter fields (see below). If *nzone* = 1, no BGR or ASC file is read and parameter fields are all constants.

PARMETER FIELD INPUT (NO BGR FILE)

```
1.  1
2.  POROSITY
3.  1  .30
4.  RETARD
5.  1   1.0
6.  LONGITUDINAL DISPERSIVITY
7.  1   0.01
8.  TRANSVE0SE DISPERSIVITY
9.  1   0.01
10. DIFFUSION COEFFICIENT
11. 1   0.000052
12. DECAY
13. 1   0.000
```

PARMETER FIELD INPUT (BGR FILE)

```
1.  2
2.  "bgrfl.bgr"
3.  POROSITY
3.  1  .20
4.  2  .30
5.  RETARD
6.  1   1.0
7.  2   1.0
8.  LONGITUDINAL DISPERSIVITY
9.  1   0.01
10. 2   0.01
11. TRANSVE0SE DISPERSIVITY
12. 1   0.01
13. 2   0.01
14. DIFFUSION COEFFICIENT
15. 1   0.000040
16. 2   0.000052
17. DECAY
18. 1   0.000
19. 2   0.000
```

2. (CHARACTER*80 *bgrfl*) If *nzone* > 1 line 2 contains the name of the TPROGS TSIM BGR ASC file.

3. Key word (CHARACTER) in capitals specifying the parameter field to be read followed by lines specifying zone number and associated parameter value (nzone ≥ 1), or the file name containing the parameter field. All key words must be specified. The key word DECAY refers to a first-order degradation-rate coefficient [T^{-1}]. For each key word, enter either **(1)** (nzone > 1) lines containing a zone number (INTEGER) and corresponding parameter value (REAL) for each zone, or **(2)** (nzone = 0) a constant (INTEGER=0,-1, or 1) followed by the unformatted (constant =-1) or formatted (constant =1) file name containing the parameter field, or a constant parameter value (REAL) for all zones (constant=0). Values in the parameter field files cycle on i, j, and then k.

Note that RWHet will attempt to read the TSIM file consecutively in the following formats until it succeeds: ASCII, unformatted binary integer*1, unformatted binary integer*4, binary integer*1, and binary integer*4. The TSIM file must have the same dimensions as the RWHet domain. The TSIM data and parameter field are internally rotated for *irevz* =1 (see the velocity field input file).

3.4 Velocity Field Input

RWHet requires a cell-by-cell unformatted binary file containing the flux field as generated by

VELOCITY FIELD INPUT

```
1.  36.5  1 1 1  1 "VEL.UFM"
2.  73.0  1 1 1 -1 "VEL.BIN"
3.  100.0 1 1 1  1 "VEL.UFM"
```

MODFLOW or similar block-centered finite difference flow code. RWHet has the option to rotate this field about the *x*-axis to transform from a MODFLOW coordinate system in which the +*z*-axis is downward to a new coordinate system in which the +*z*-axis is upward. Each line of the velocity input file contains (1) (REAL *tnextvel*) the time to read the next velocity field, (2) (INTEGER *kstp*) MODFLOW time step (see the MODFLOW documentation), (3) (INTEGER *kper*) MODFLOW stress period (see MODFLOW documentation), (4) (INTEGER *irevz*) a flag specifiying if the coordinate system is to be transformed (*irevz* = 1) or not (*irevz* = 0); (ivtype) velocity file type either unformatted binary (ivtype<0) or binary (ivtype=1), (CHARACTER vfile) name of cell-by-cell flux file containing MODFLOW formatted fluxes.

3.5 Internal Boundary Condition Input

RWHet contains 9 internal boundary types: constant concentration (*bc_type*=1), continuous mass input (*bc_type*=2), absorbing (*bc_type*=3), absorbing with specified flux (*bc_type*=4), absorbing with flux computed from the MODFLOW cell-by-cell flux field (*bc_type*=5), an injection well (*bc_type*=6), a multi-aquifer absorbing well (*bc_type*=7), specified mass flux in recharge (*bc_type*=8), and specified concentration in recharge (*bc_type*=9) as described in the following sections. Boundaries can be grouped together to reduce output and average results, for example, over a well screened in multiple intervals.

3.5.1 Constant concentration (*bc_type* = 1)

INTERNAL BOUNDARY CONDITION INPUT

```
# boundary type 1 at i,j,kbot,ktop = 20,20,30,60
# 1000 particles, conc=1.0, refresh=1.0,
# tbeg=0.0, tend=365.0
1.   1 20,20,30,60,1000,1.0,1.0,0.0,365.0
```

A constant concentration boundary can be maintained in a cell by placing particles of appropriate mass in that cell with uniform distribution, and periodically refreshing this distribution by absorbing all particles within the boundary cell, and then instantly redistributing a new set of particles. Input for boundary type 1 includes (1) (INTEGER *bc_type* = 1) the boundary type, (2) (INTEGER *i, j, kbot,ktop*) cell locations (vertical interval) at which it is applied, (3) (INTEGER *npart*) number of particles to distribute each time the boundary is refreshed, (4) (REAL *conc*) concentration [ML^{-3}], (5) (REAL refresh) time interval on which the boundary is

refreshed, (6) (REAL *tbeg*) beginning time at which the boundary becomes active, (7) (REAL *tend*) ending time at which the boundary becomes inactive, (8) (INTEGER *group*) boundary group number (must occur consecutively). Internal counters for boundary conditions in the same group are reported as a single total in output.

3.5.2 Continuous mass input (*bc_type* = 2)

A continuous source can be specified in a region by distributing, with uniform distribution, particles at a specified rate and mass. This approach can also be used to maintain a constant concentration if the flux of water is known. Input for boundary type 2 includes (1) (INTEGER *bc_type* = 2) the boundary type, (2) (REAL *xc*, *yc*, *zc*) location of the center of the region, (3) (REAL sx, sy, sz) size of theregion in the x, y, and z directions, (4) (REAL *pmass*) mass to assign to each particle, (5) (REAL *massrate*) mass per unit time [MT^{-1}] entering the boundary region, (6) (REAL *tbeg*) time at which the boundary becomes active, (7) (REAL *tend*) time at which the boundary becomes inactive, (8) (INTEGER *group*) boundary group number, must occur consecutively.

3.5.3 Absorbing (*bc_type* = 3)

Particles that enter absorbing boundaries are removed from the simulation. Input for boundary type 3 includes (1) (INTEGER *bc_type* = 3) the boundary type, (2) (INTEGER *i, j, kbot, ktop*) cell location at which it is applied, (4) (REAL *tbeg*) time at which the boundary becomes active, (5) (REAL *tend*) time at which the boundary becomes inactive, (6) (INTEGER *group*) boundary group number, must occur consecutively.

3.5.4 Absorbing with specified flux (*bc_type* = 4)

An absorbing boundary with specified flux removes particles at each time step with probability, equal to the specified volumetric flow rate Q times the time step Δt divided by the total volume of water in (scaled by R), and leaving, that boundary cell during the time step, computed as $Q\Delta t/(Q\Delta t+R\Theta\Delta x\Delta y\Delta z)$. If the flux is greater than this volume, this boundary condition implicitly reverts to a boundary type 3, i.e., the probability is limited to maximum of 1. Input for boundary type 4 includes (1) (INTEGER *bc_type* = 4) the boundary type, (2) (INTEGER *i, j, kbot, ktop*) cell location at which it is applied, (3) (REAL *wflux*) specified flux [L^3T^{-1}], (4) (REAL *tbeg*) time at which the boundary

becomes active, (5) (REAL *tend*) ime at which the boundary becomes inactive, (6) (INTEGER *group*) boundary group number, must occur consecutively.

3.5.5 Absorbing with computed flux (*bc_type* = 5)

An absorbing boundary with computed flux removes particles with probability computed as above, but with the specified volumetric flow rate *Q* computed from the divergence of the flux field in the boundary cell. Input for boundary type 5 includes (1) (INTEGER *bc_type* = 5) the boundary type, (2) (INTEGER *i, j, kbot, ktop*) cell location at which it is applied, (3) (REAL *tbeg*) time at which the boundary becomes active, (4) (REAL *tend*) time at which the boundary becomes inactive, (5) (INTEGER *group*) boundary group number.

3.5.6 Injection Well (*bc_type* = 6)

An injection well is simulated by injecting mass according to (a) the local flux at the edges of the specified vertical column, (b) the specified mass-rate of injection, and the specified particle mass. Input for boundary type 6 includes (1) (INTEGER *bc_type* = 6) the boundary type, (2) (INTEGER *i, j, kbot,ktop*) cell locations (vertical interval or well) at which it is applied, (3) (REAL *pmass*) particle mass, (4) (REAL *massrate*) rate of mass injection [MT^{-T}], (5) (REAL *tbeg*) beginning time at which the boundary becomes active, (6) (REAL *tend*) ending time at which the boundary becomes inactive, (8) (INTEGER *group*) boundary group number (must occur consecutively). Internal counters for boundary conditions in the same group are reported as a single total.

3.5.7 Multi-aquifer absorbing with computed flux (*bc_type* = 7)

A multi-aquifer well is simulated by capturing particles at a well (using the method described for the absorbing boundary, *bc_type=5*) and injecting mass (using the method described for *bc_type=6*), if there is flow from the well and back into the system. Concentration associated with the injected mass is based upon an assumption of complete mixing in the well bore (i.e., the concentration everywhere in the well is given by the mass captured divided the volume of water pumped). Input for boundary type 7 includes (1) (INTEGER *bc_type* = 7) the boundary type, (2) (INTEGER *i, j, kbot,ktop*) cell locations of the well at which it is applied, (4) (REAL *tbeg*) beginning time at which the

boundary becomes active, (5) (REAL *tend*) ending time at which the boundary becomes inactive, (6) (INTEGER *group*) boundary group number (must occur consecutively). Internal counters for boundary conditions in the same group are reported as a single total in output.

3.5.8 Type 1 recharge boundary (*bc_type* = 8)

The recharge (type 1) boundary is a continuous source in an area at the water table maintained by distributing, with uniform distribution, particles at a specified rate and mass. Input for boundary type 8 includes (1) (INTEGER *bc_type* = 8) the boundary type, (2) (REAL *xc*, *yc*) location of the center of the region, (3) (REAL sx, sy) size of theregion in the x and y directions, (4) (REAL *pmass*) mass to assign to each particle, (5) (REAL *massrate*) mass per unit time [MT^{-1}] entering the boundary region, (6) (REAL *tbeg*) time at which the boundary becomes active, (7) (REAL *tend*) time at which the boundary becomes inactive, (8) (INTEGER *group*) boundary group number, must occur consecutively. This boundary type requires a recharge flux within the boundary region, as well as recharge information in the cell-by-cell flux file. Mass will not be distributed in cells of the specified region where the recharge flux is less than or equal to zero. Specification of a recharge flux in an area with no positive recharge flux will result in errors.

3.5.9 Type 2 recharge boundary (*bc_type* = 9)

The recharge type 2 boundary is a continuous source specified in a rectangular area at the water table. The algorithm distributes particles with masses computed to maintain uniform concentration within the specified boundary region. Input for boundary type 9 includes (1) (INTEGER *bc_type* = 9) the boundary type, (2) (INTEGER *im*, *jm*) *i*,*j* location of the start of the boundary region, (3) (INTEGER *ip*, *jp*) *i*,*j* location of the end of the boundary region in the *x*- and *y*- directions, (4) (REAL *conc*) concentration [ML^3] entering the boundary region, (5) (REAL *nptime*) number of particles per unit time to release within each *i*,*j* location, (6) (REAL *tbeg*) time at which the boundary becomes active, (7) (REAL *tend*) time at which the boundary becomes inactive, (8) (INTEGER *group*) boundary group number, must occur consecutively. Implementing this boundary type requires a positive recharge flux within at least 1 cell in the boundary region, as well

as recharge information in the cell-by-cell flux file. Mass will not be distributed in cells of the specified region where the recharge flux is less than or equal to zero. Specification of a recharge flux in an area with no positive recharge flux will result in errors.

3.5.10 Constant concentration inflow (*bc_type* = 10)

The constant concentration inflow is a continuous source specified in a rectilinear volume for which a specified source of water is contained in the flow field input (e.g., a constant head or general head boundary flux). The algorithm distributes particles uniformly with masses computed to maintain uniform concentration within the specified boundary region. Input for boundary type 10 includes (1) (INTEGER *bc_type* = 10) the boundary type, (2) (INTEGER *im, jm,km*) *i,j,k* location of the start of the boundary region, (3) (INTEGER *ip, jp,kp*) *i,j,k* location of the end of the boundary region in the *x*- and *y*-directions, (4) (REAL *conc*) concentration [ML^3] entering the boundary region, (5) (REAL *nptime*) number of particles per unit time to release within each *i,j,k* location, (6) (REAL *tbeg*) time at which the boundary becomes active, (7) (REAL *tend*) time at which the boundary becomes inactive, (8) (INTEGER *group*) boundary group number, must occur consecutively. Implementing this boundary type requires a flux or water to the domain within at least 1 cell in the boundary region, as well as the corresponding information in the cell-by-cell flux file. Mass will not be distributed in cells of the specified region where the flux is less than or equal to zero.

3.5.11 Constant concentration on box (*bc_type* = 11)

The constant concentration on box boundary is a continuous source specified on the edges of a rectilinear volume. Unlike boundary type 10 described above, type 11 does not require a specified source of water be contained in the flow field input. Instead, the algorithm distributes particles uniformly with masses computed to maintain uniform concentration in the flow leaving the specified boundary region. Thus, particles are only placed on cell faces with an outward flux. Input for boundary type 11 includes (1) (INTEGER *bc_type* = 11) the boundary type, (2) (INTEGER *im, jm,km*) *i,j,k* location of the start of the boundary region, (3) (INTEGER *ip, jp,kp*) *i,j,k* location of the end of the boundary region in the *x*- and *y*- directions, (4) (REAL *conc*) concentration [ML^3] entering the boundary region, (5) (REAL *nptime*) number of particles per unit time to

release within each *i*,*j*,*k* location, (6) (REAL *tbeg*) time at which the boundary becomes active, (7) (REAL *tend*) time at which the boundary becomes inactive, (8) (INTEGER *group*) boundary group number, must occur consecutively. Note that the specified concentration is converted to a mass input without taking into account retardation. Thus, the resulting aqueous concentration may be reduced for a retardation factor greater than one.

3.5.12 Point Particle Input

POINT PARTICLE INPUT

```
# tnextpnt, number of lines of input
0.0000 3
# x, y, z, number of particles, particle mass
1.0 1.0 1.0 1000 1.0
1.0 1.0 2.0 1000 1.0
1.0 1.0 3.0 1000 1.0
# tnextpnt, number of lines of input
365.00 1
# x, y, z, number of particles, particle mass
1.0 1.0 1.0 6000 1.0
```

This input file is used to specify particle input at specified points and times.

1. (REAL *tnextpnt*, INTEGER *npntsrc*) Time to initialize new particles *tnextpnt* and the number of lines of input in this set of points.

2. (REAL *x*,*y*,*z* INTEGER *nppnt* REAL *pmass*) *x*,*y* , *z* location of point, number of particles *nppnt* to distribute at that point, mass of each particle *pmass*. The example specifies three instantaneous point sources of 1000 partcles of mass 1.0 at time 0.0, and one source of 6000 particles of mass 1.0 at time 365.0.

3.5.13 Sampling Input

SAMPLING INPUT

```
# itype = 3, x-z plane at location 5,
# max conc. in zone 4
3 5 1 4
# itype = 2, y-z plane at location 6,
# mass in zones 3 and 4
2 6 2 -3 -4
# cylinder at x,y = 1,1 zbot,ztop = 2,3, radius = 4,
# concentration in zones 1 and 2
1 1.0 1.0 2.0 3.0 4.0 2 1 2
```

This input file (*.sam) can be used to monitor concentrations at specific points within the domain. Options are currently available to monitor average concentration with a vertical circular cylinder (itype = 1), on a plane of grid blocks oriented perpendicular to the x, y or z planes (itype = 2, 3, or 4), in a column of grid blocks (itype = 5), in a domain on either side of a plane oriented perpendicular to either the x, y or z direction (itype = 6, 7, or 8). Monitoring within specific TPROGS zones is accomplished by specifying the number of zones (sam%nzone) and specific zones to monitor (sam%nzone = 0 for all zones).

<u>For itype =1</u>
itype,x,y,zbot,ztop,radius,sam%nzone,nzone zone numbers

For itype =2, 3, 4
itype, i (itype=2), j (itype=3), k (itype=4),sam%nzone,nzone zone numbers (if positive output reports maximum concentration in plane, if negative output reports total mass)

For itype =5
itype,i,j,kbot,ktop, sam%nzone,nzone zone numbers,sam%nzone,nzone zone numbers

For itype =6, 7, 8
itype, i (itype=6), j (itype=7), k (itype=8),sam%nzone,nzone zone numbers (if positive, output reports maximum concentration in plane, if negative output reports total mass). If *i, j,* or *k* is negative, then the region of the domain monitored is less than or equal to the absolute value of the specified number; if *i, j,* or *k* is positive, then the region of the domain monitored is greater than or equal to the specified number.

3.6 Model Ouput

3.6.1 Spatial Moments

Spatial moments are output in standard GEOEAS format [please refer to the TPROGS manual; Carle and Fogg, 1998].

SPATIAL MOMENTS

```
spatial moments
        10
time
first spatial moment x-direction
first spatial moment y-direction
first spatial moment z-direction
covariance xx
covariance xy
covariance xz
covariance yy
covariance yz
covariance zz
 0.0000 207.4920  25.0011  21.5006   2.0815  -0.0168  -0.0022   8.3713  -0.0109   0.3338
36.5000 207.4918  25.7636  21.5010   2.0885  -0.0124  -0.0026   8.4195  -0.0080   0.3425
      .       .       .       .       .       .       .       .       .       .
      .       .       .       .       .       .       .       .       .       .
      .       .       .       .       .       .       .       .       .       .
```

3.6.2 Concetrations

Concentrations in a cell are computed from the sum of particle masses m_p contained in that cell divided by the volume of water in that cell times the retardation coefficient $c_{ijk} = \sum_{p \in V_{ijk}} m_p \Big/ \left(R\Theta\Delta V \right)$. Output can be specified as the number of particles in a cell (opc(2) = 1), as RWHet formatted concentrations (opc(2) = 2), or as both RWHet and MT3DMS formatted concentrations (opc(2) = 3; see the MT3DMS manual, Zheng, 2006). The MT3DMS output files always

CONCENTRATION OUTPUT (BINARY)

```
1.  nx,ny,nz,dx,dy,dz
2.  curtime,ncl
3.  ncell1
4.  cell2
5.  curtime,ncl
6.  ncell1
7.  cell2
      .
      .
      .
```

have the name of the RWHet concentration file (e.g., rwhet.con) with the extension changed by adding "_001.ucn" (e.g., rwhet_001.ucn for true binary formatted concentrations) and ".cnf" (e.g., rwhet.cnf for the MT3DMS definition file). The RWHet format for the concentration file is binary:

1. (INTEGER nx, ny, nz, REAL, dx, dy, dz) Model dimensions and discretization.

2. (REAL *curtime*, INTEGER *ncl*) Current time and number of cells containing particles.

3. (INTEGER ncel11) Array of dimension *ncl* containing locations of all cells containing particles computed as $i+(j-1)*nx+(k-1)*nx*ny$, where i, j, and k index cells in the x, y and z directions. The coordinate system for all output is right handed with z positive upward (irevz=0), or z positive downward (irevz=1; see Section 3.4).

4. (INTEGER *ncel12* or REAL *cell2*) Array of dimension *ncl* containing either integer values specifying the number of particles in each cell (opc(2) = 1) or real values specifying concentrations (opc(2) = 2 or 3). Locations or the associated output correspond to those specified in ncell1 in the same order. Records 2 – 4 are repeated for all times corresponding to output as specified by opc(2).

3.6.3 Breakthrough

```
                    BREAKTHROUGH OUTPUT
1.        0.0000  80000  80000    0    0    0    0    0    0
2.      365.0000  80217    217    0    0    0    0    0    0
        :
        :
        :
```

Breakthrough output can be specified in numbers of particles (opc(4) = 1) or mass (opc(4) = 2). The ASCII output contains the (1) current time followed by (2) the total number of particles *np* or total mass, (3) the incremental change in this total since the last output to this file, and (4) 6 values corresponding to numbers of particles or total mass passing through planes at the extent of the active model domain in the $+x$, $-x$, $+y$,$-y$, $+z$, and $-z$ directions.

3.6.4 Particle Attributes

```
                    PARTICLE ATTRIBUTE OUTPUT
1.   80000 0.0
2.   1  1.0000  1.0000  1.0000  21.5006  207.4920  25.0011  21.5006  0.0  1.0
       :
       :
       :
```

Particle attribute output for each time where opc(5) = 1 includes the number of particles *np* and current time *curtime* on the first line

followed by *np* lines of output, one for each particle, showing particle (1) number (integer), (2) current location (real) *x*, *y* , *z*, (3) age (real; the current time minus the time that the particle was born, i.e., entered the simulation), and (4) mass (double precision real) m_p. Particle attribute output is binary unformatted and can be converted to ASCII using the prtreader.f90 utility provided with RWHet.

3.6.5 Internal Boundary Condition Particle and Mass Counters

```
      INTERNAL BC PARTICLE AND MASS OUTPUT
0.0000,       -1,       1,   80000,       4
0.0000,        5,       2,       0,       5
36.5000,      -1,       1,       0,       4
36.5000,       5,       2,       0,       5
73.0000,      -1,       1,       0,       4
73.0000,       5,       2,       0,       5
       .
       .
       .
```

Internal boundary condition particle and mass counters track the total number of particles and mass entering and leaving specified internal boundaries. RWHet writes one line of output at specified times for each boundary group. Output for each line includes the (1) current time *curtime*, (2) boundary type (negative if currently inactive) *bc_type*, (3) boundary group number *group*, (4) totals entering (+) or leaving (-) the boundary group (from the previous output to this file) given in either numbers of particles (opc(6) = 1) or mass (opc(6) = 2), (5) the mass transferred or relocated in multi-aquifer well (MAW) boundaries (m_{MAW}), (6) the flux f_{in} [L^3T^{-1}] heading into a MAW boundary (includes sum of pumpage and flow into the system), (7) the flux f_{out} [L^3T^{-1}] heading out of a MAW boundary (flow into the system), and (8) the number of individual boundary conditions included in this group. Note that the concentration in a MAW boundaries can be computing as $m_{MAW}/\min(|f_{out}|, f_{in})$. If the boundary group contains only one boundary condition then each line of output for that group also contains the *i*, *j*, *kbot*, *ktop* location of the boundary if applicable (i.e., if the boundary is specified by cell location, e.g., *bc_type* = 1).

3.6.6 Monitoring File

Provides results for each monitoring point specified in the sampling input (*.sam).

3.6.7 Particle Removal Output

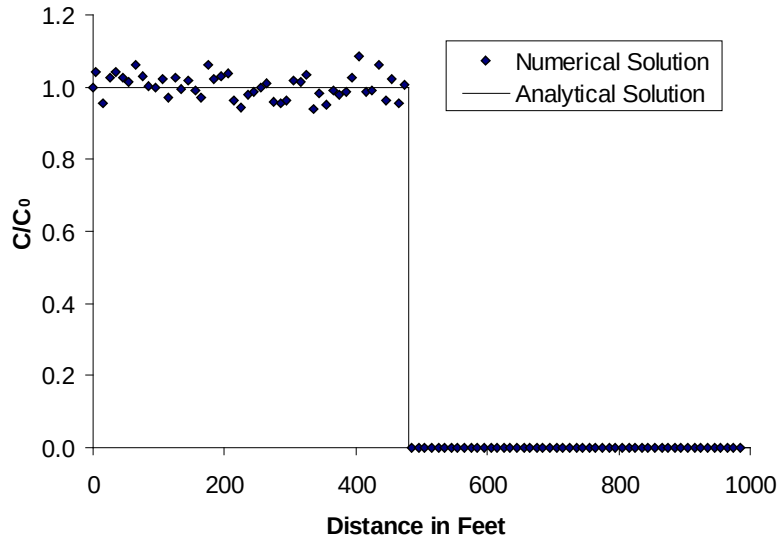
```
      PARTICLE REMOVEAL OUTPUT (BINARY)
1.   -1
2.   1.0 1.0 1.0
3.   1
4.   0.0 365.4 2.0 2.1 1.1 1.0D+01 2 2 1
   .
   .
   .
```

Attributes of particles removed during simulation are logged in the second output file specified during execution of the program. The format is binary. The starting location for particles are logged as a negative particle number (1 - INTEGER) and on the following line a (2 - REAL) starting location, *x*, *y*, *z*. For a particle removed, one line is written containing (1 - INTEGER) particle number, followed by another containing (2 - REAL) starting time, (3 - REAL) ending time, (4 - REAL) ending location *x*, *y* , *z*, (5 – DOUBLE PRECISION REAL) mass m_p, and (6 - INTEGERS) ending cell *i*,*j*,*k*.

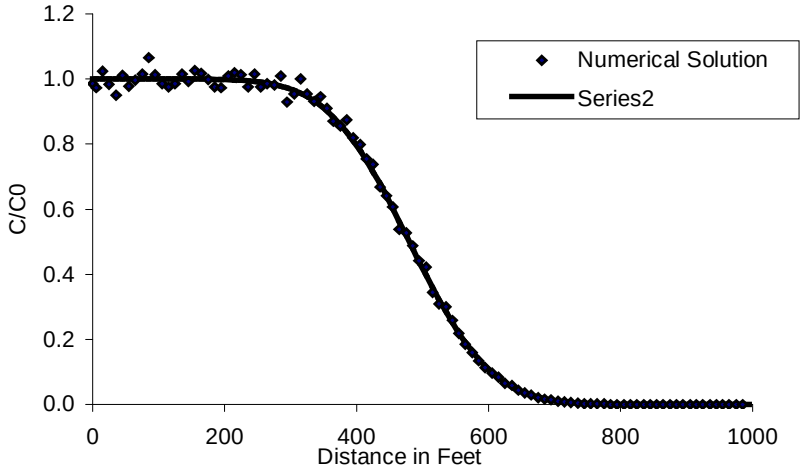
4.0 Example Results

4.1 Example 1

Example 1 considers one-dimensional solute transport involving advection and dispersion in a steady-state uniform flow field. Details for this problem are discussed in MT3D [1991, p7-1]. In Case 1, there is advection only with a groundwater seepage velocity of 0.24 ft/day. Case 2 includes dispersion and advection. The analytical solution (Van Genuchten and Alves, 1982) was calculated from the program ODAST (Javandel et al. 1995, p129-132).



(a) Case 1: without sorption, decay, and dispersion

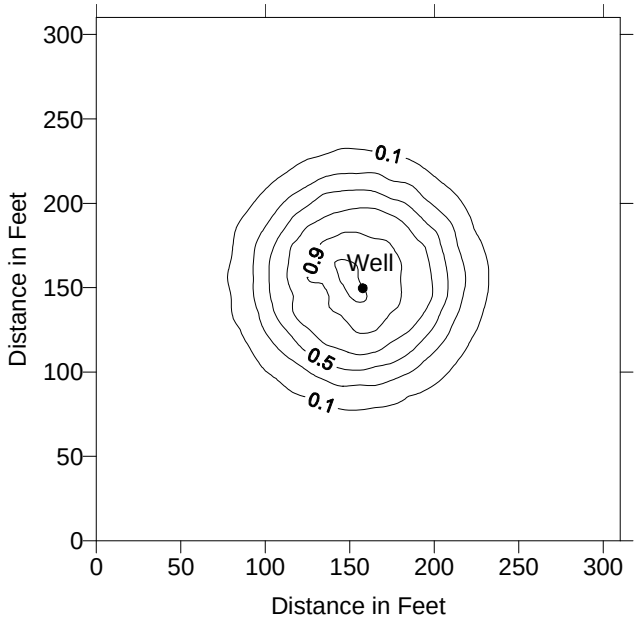


(b) Case 2: with dispersion, without sorption and decay

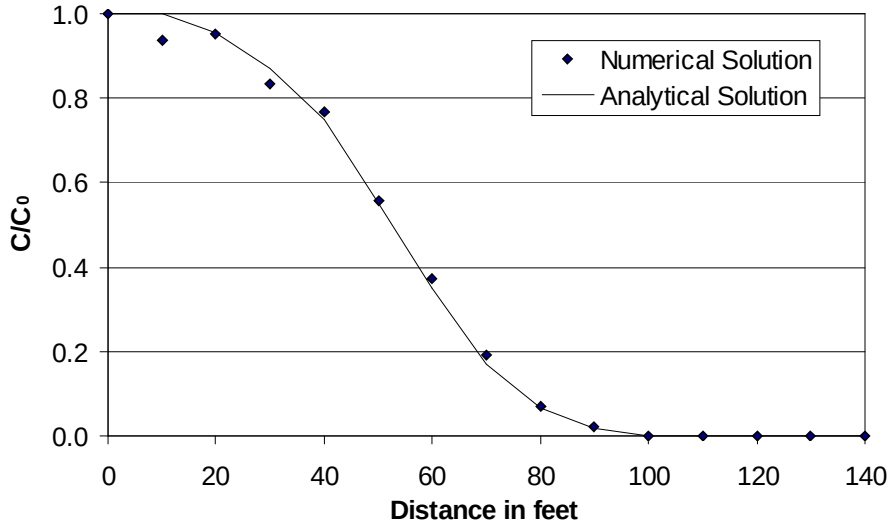
Figure 1. Breakthrough curves for the one-dimensional test problem are compared with analytical solutions.

4.2 Example 2

Example 2 simulates the two-dimensional transport of solute injected from a fully penetrating well (Figure 3a). Details for this problem are discussed in MT3D (1991, p7-9). The analytical solution was calculated using the program LTIRD (Javandel et al. 1995, p167-170).



(a) Contour map of the concentration field as calculated by rwhet2.0



(b) Breakthrough curve along the *x*-axis

Figure 2. Comparison of numerical and analytical solutions for two-dimensional transport from a point source in a radial flow field.

4.3 Example 3

Example 3 involves an injection/pumping cycle for a fully penetrating well in a confined aquifer . Details for this problem are discussed in MT3D (1991, p10-11). The analytical solution is given in Gelhar and Collins (1991).

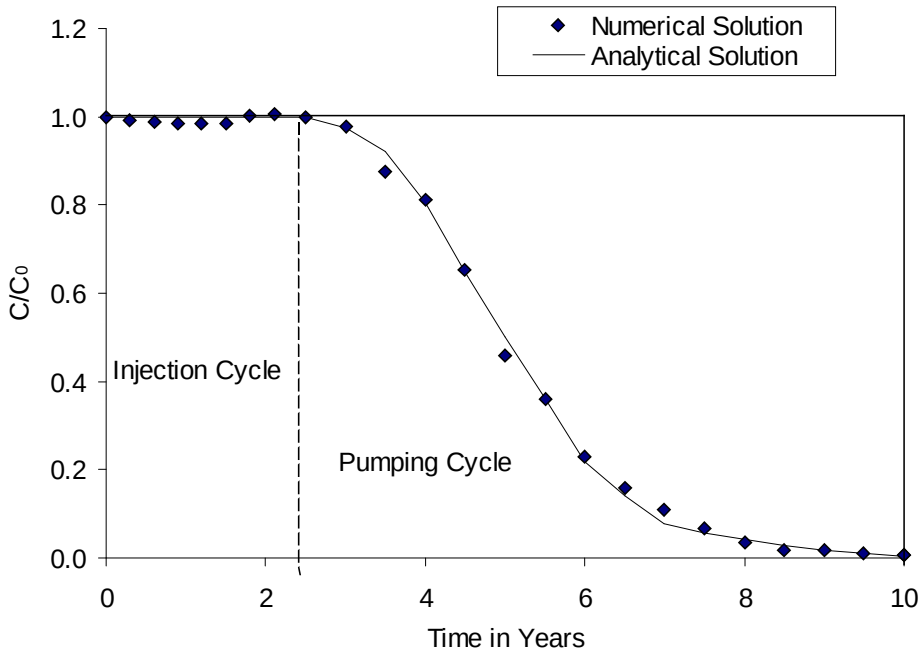


Figure 3. RWHet2.0 solution is compared with the analytical solution of Gelhar and Collins [1971] for relative concentration at a well during an injection/pumping cycle.

4.4 Example 4

Example 4 shows results from a simulation of conservative transport in the alluvial fans of the Lawrence Livermore National Laboratory (LLNL). Details regarding the associated 3-D simulation of facies architecture can be found in *Carle et. al.* [1998]. Details regarding the transport simulation are available in *LaBolle* [1999].

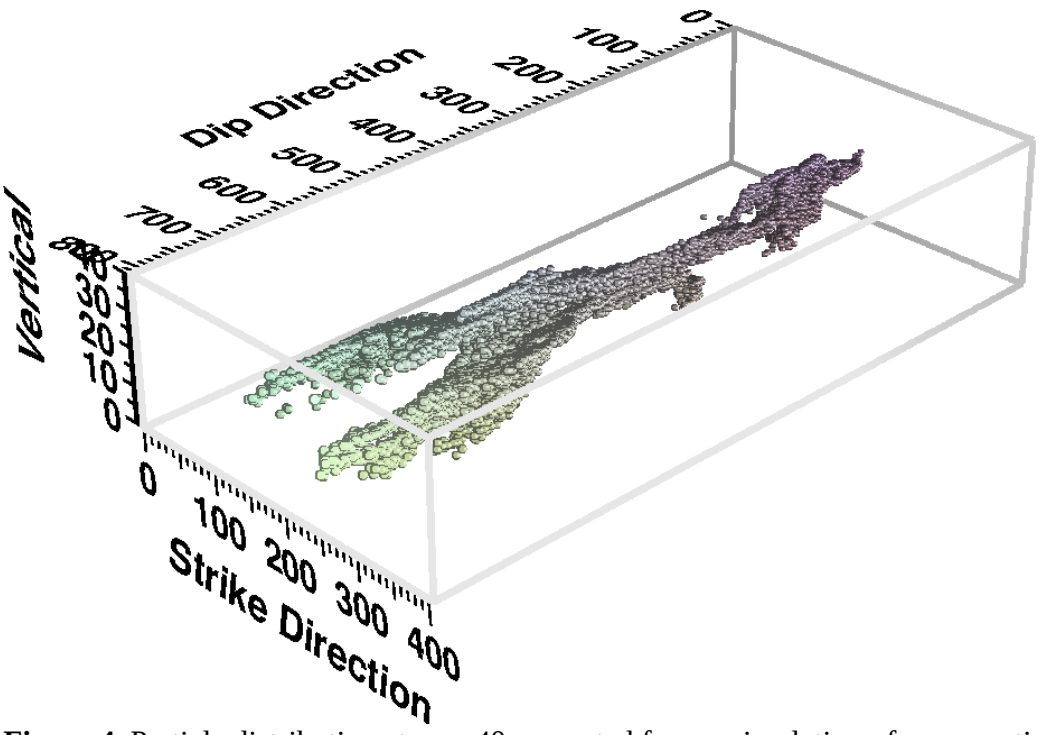
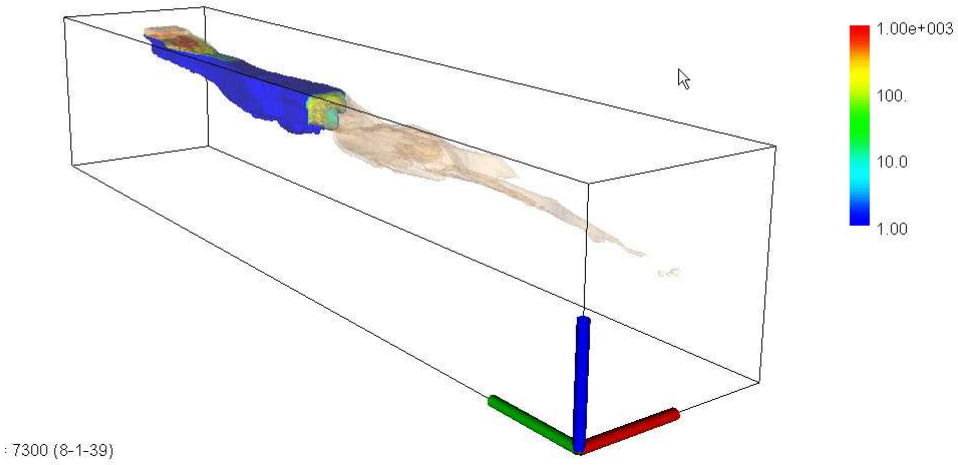


Figure 4. Particle distribution at year 40 generated from a simulation of conservative transport in the LLNL alluvial-fan system.

4.5 Example 5

Example 5 shows results from a simulation of conservative transport in alluvial fans of the eastern Central Valley. Results are shown for a 5,000,000 version of the 3-D simulation of facies architecture found in *Weissmann et al.* (2002). The source is a transient, and distributed over 40 acres through recharge.

(a)



(b)

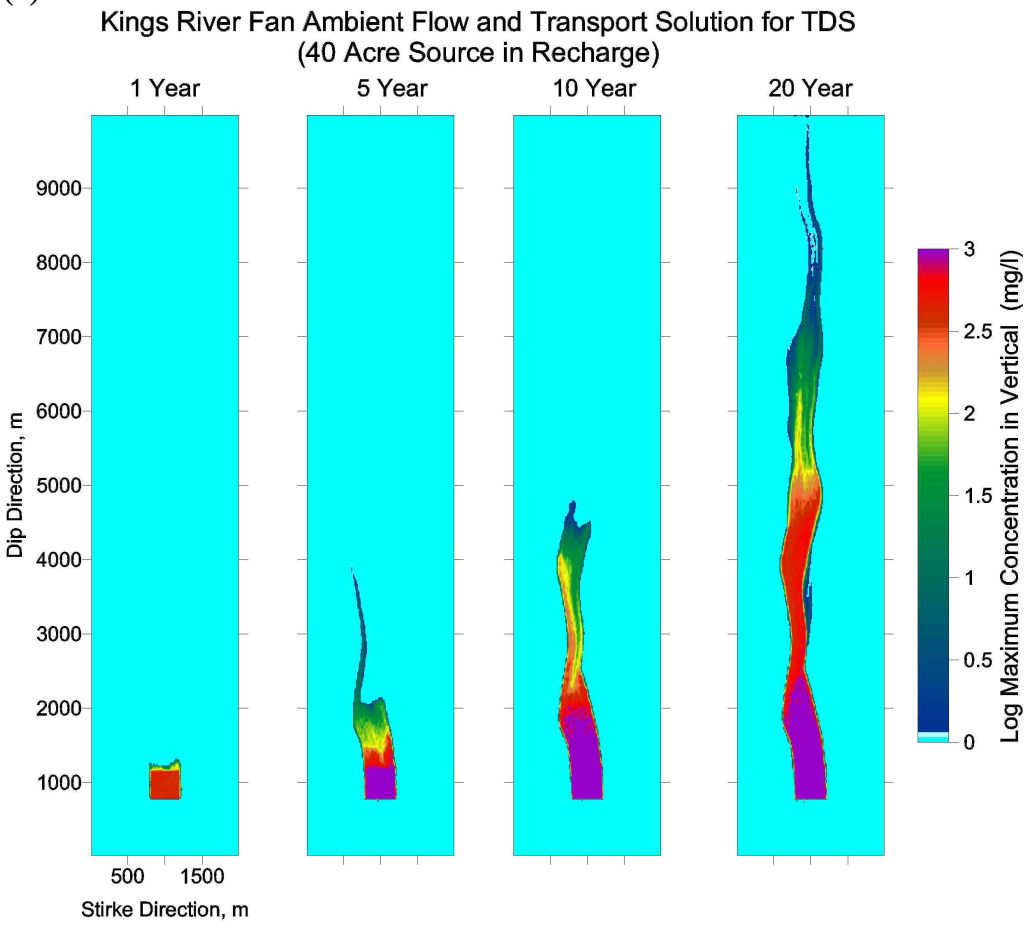


Figure 5. (a) Results from a transient release of elevated TDS concentrations over 20 years in 40 acres of seasonally varying recharge; graphics were generated using the free USGS model viewer code and the MT3D output file format generated by RWHet, and (b) time evolution of the maximum concentration in the vertical. Concentrations start low and increase to about 1,500 mg/l with some seasonal variation.

5.0 References

Bagtzoglou, A.C., A.F.B. Tompson and D.E. Dougherty, Projection functions for particle grid methods, *Numerical Methods for Partial Differential Equations*, 8, 325-340, 1992.

Carle, S.F., E.M. LaBolle, G.S. Weissmann, D. VanBrocklin, and G.E. Fogg, Geostatistical simulation of hydrofacies architecture: A transition probability/Markov approach, in SEPM Concepts in Hydrogeology and Environmental Geology No. 1, Hydrogeologic Models of Sedimentary Aquifers, G.S. Fraser and J.M. Davis (eds), SEPM (Society for Sedimentary Geology), Tulsa Oklahoma, 1998.

Carle, S.F. and G.E. Fogg, TPMOD: A transition probability/Markov approach to geostatistical modeling and simulation, University of California, Davis, 1998.

Zheng, C., *MT3DMS v5.2 Supplemental User's Guide*, Technical Report to the U.S. Army Engineer Research and Development Center, Department of Geological Sciences, University of Alabama, 24 p, 2006.

Fogg, G.E., E.M. LaBolle, G.S. Weissmann, Groundwater vulnerability assessment: hydrogeologic perspective and example from Salinas Valley, CA, in *Application of GIS, Remote Sensing, Geostatistical and Solute Transport Modeling to the Assessment of Nonpoint Source Pollution in the Vadose Zone, AGU Monograph Series 108*, 1998.

Gardiner, C.W., *Handbook of Stochastic Methods for Physics Chemistry and the Natural Sciences*, Springer-Verlag, Berlin, 1990.

Gelhar, L. W., and M. A. Collins, 1971. General analysis of longitudinal dispersion in uniform flow

Javandel, J., C. Doughty & C. E. Tsang, Groundwater Transport: Handbook of Mathematical Models, 1995, p129~132.

Kinzelbach, W., The random walk method in pollutant transport simulation, in *Groundwater Flow and Quality Modelling,* ed. Custidio, E., A. Gurgui, and J.P. Lobo Ferreria, NATO ASI Series C: Math and Phys. Sci. 224: 227-246, Reidel Publishing Company, 1988.

LaBolle, E.M., Theory and Simulation of Diffusion Processes in Porous Media, Dissertation, University of California, Davis, 202 p., 1999.

LaBolle, .E.M., J. Quastel, G.E. Fogg, and J. Gravner, Diffusion processes in composite porous media and their numerical integration by random walks: Generalized stochastic differential equations with discontinuous coefficients, *Water Resources Research, 36(3), 651-662, 2000.*

LaBolle, E.M., G.E. Fogg, and A.F.B. Tompson, Random-Walk Simulation of Transport in Heterogeneous Porous Media: Local Mass-Conservation Problem and Implementation Methods, *Water Resources Research, 32(3)*, 583-593, 1996.

LaBolle, E.M., J. Quastel, and G.E. Fogg, Diffusion theory for transport in porous media: Transition-probability densities of diffusion processes corresponding to advection-dispersion equations, *Water Resources Research, 34(7),* 1685-1693, 1998.

LaBolle, E.M. and Y. Zhang, Reply to Comment by Doo-Hyun Lim on "Diffusion processes in composite porous media and their numerical integration by random walks: Generalized stochastic differential equations with discontinuous coefficients", Water Resour. Res.,42(2), 2006.

McDonald, M.G. and Harbaugh, A.W, A modular three-dimensional finite-difference ground-water flow model. U.S. Geological Survey Techniques of Water-Resources Investigations Book 6, Chapter A1, 586 p., 1988.

MT3D, A Modular Three-Dimensional Transport Model for the Simulation of Advection, Dispersion and Chemical Reactions of Contaminants in Groundwater Systems, S.S. Papadopulous & Associates, Inc., 1991.

Pollock, D.W., Semianalytical computation of path lines for finite-difference models, *Groundwater, 26(6),* 743-750, 1988.

Risken, H., *The Fokker-Planck Equation,* Springer Verlag, Berlin, 1989.

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

Tompson, A.F.B., E.G. Vomoris, and L.W. Gelhar, Numerical simulation of solute transport in randomly heterogeneous porous media: motivation, model development, and application, Report UCID-21281, Lawrence Livermore National Laboratory, 1987.

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

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

Uffink, G.J.M., Modeling of solute transport with the random walk method, in *Groundwater Flow and Quality Modelling,* ed. Custidio, E., A. Gurgui, and J.P. Lobo Ferreria, NATO ASI Series C: Math and Phys. Sci. 224: 247-265, Reidel Publishing Company. 1988.

Van Genuchten, M. Th. & W. J. Alves. 1982. Analytical solutions of the one-dimensional convective-dispersive solute transport equation. U.S. Department of Agriculture Technical Bulletin No. 1661, p151.

Weissmann, G., Y. Zhang, E.M. LaBolle and G. E. Fogg, Dispersion of Groundwater Age in an Alluvial Aquifer, Water Resources Research, Vol. 38, No. 10 , p. 1198, 2002.