

PPTTEST:
A Multi-Species Reactive Transport
Model to Estimate Biogeochemical Rates
Based on Single-Well Push-Pull Test Data
User Manual

**Report Number:
2010ENE-01**

**Mantha S. Phanikumar
Department of Civil & Environmental Engineering
Michigan State University, East Lansing, MI 48824, USA
Email: phani@msu.edu**

Summer 2010

PPTTEST: User Manual

Version 2.0

July 23, 2010

Errata

This user manual is written to describe the PPTTEST model (Phanikumar and McGuire, 2010) in more detail . In the original paper, two typographical errors were not corrected at the proofs stage. These are described below.

1. In equation (12) on page 999, the last term should appear as shown below:

$$+ \dots + \left\{ \begin{array}{ll} k_{N-1} C_i^{n_{N-1}} & t_{N-1}^* \leq t < t_N^* \\ 0 & \text{otherwise} \end{array} \right\}$$

2. On page 999 (below equation 1), the sentence "N is the total number of reaction constants" should read " (N - 1) is the total number of reaction constants".

Phanikumar, M.S. and J.T. McGuire, A multi-species reactive transport model to estimate biogeochemical rates based on single-well push-pull test data, *Computers & Geosciences*, Vol. 36, No. 8, pp. 997-1004 (August 2010)

1 Introduction

PPTEST is a one-dimensional, radial-coordinate, finite-difference model to analyze data obtained from a variety of single-well push-pull tests. The model allows the simulation of multiple species, arbitrary reaction order kinetics and multiple reaction rates to describe the observed data. A number of sorption models have been built into the model and additional user-defined models can be easily incorporated into the code. This document describes the theory and the numerical details and provides examples to illustrate the application of PPTEST. This document is organized as follows. In the next section, we present the governing equations, boundary and initial conditions. In section 3 details of the computational meshes and numerical schemes are discussed. PPTEST prompts the user for a text file that contains the input parameters. In section 4 we describe the input parameters. Examples and sample input files are presented in section 5. The program listing is given in section 6. The source code can be compiled using any FORTRAN compiler although results reported in the paper were obtained using the Intel Fortran Compiler version 11. The code was compiled using the following command:

```
>> ifort ppt.f90
```

For the data and analyses reported in this manual, PPTEST took less than a minute to run - typical running times are a few seconds. Performance can be further improved using high-level / processor-specific optimizations (e.g., Intel, 2008). Since the present method of analysis relies on numerical solutions (as opposed to analytical solutions), two points are probably worth mentioning. First, care should be exercised to ensure that the numerical solutions are accurate and do not suffer from excessive numerical dispersion since the schemes used are second-order accurate at best. By refining the grid and rerunning PPTEST, one can get an idea about the errors involved and the resolution required for a given problem. The second point is related to the input parameter `idiag` which controls the node number for which concentration versus time information is generated (e.g., for comparison with observed data). It is important to keep in mind that when the number of grid points changes, `idiag` corresponding to a radial coordinate r also changes. Therefore, to compare solutions obtained using two different grids at the same radial location, different `idiag` values should be used. When the program terminates successfully, several output files will be created - a `mesh.txt` with three columns for j , Δr_j and r_j and files that contain the time versus concentration information (the number of files created is equal to the number of species simulated). Filenames have the form `out1.dat`, `out2.dat`. The `mesh.txt` file is useful for understanding how the `idiag` value changes as the mesh-related parameters are changed. PPTEST can handle up to a maximum of 10001 grid points, 10 species and 10 reaction constants without requiring any changes to the source code. To increase these numbers, the `parameter` statements on lines 8, 95, 460, 514, 583, 599, 616 and 647 should be changed and the code recompiled.



Figure 1: Schematic showing the different phases in a typical push-pull test.

2 Governing Equations

PPTEST can solve an arbitrary number of partial differential equations (similar to equations 1 and 2 below) for tracer (denoted by $c(1,i)$ in the program) and reactive species ($c(2,i)$, $c(3,i)$, \dots in the code). Symbols are explained in the nomenclature section at the end of the manual. The equations can have complex source and sink terms and coupled to other equations (e.g., to describe microbial processes such as growth on a substrate etc). Equation (1) for the tracer is a special case of equation (2), therefore we focus on equation (2) for the i -th species.

$$\frac{\partial C_T}{\partial t} + v \frac{\partial C_T}{\partial r} = \alpha_L |v| \frac{\partial^2 C_T}{\partial r^2} \quad (1)$$

$$\frac{\partial C_i}{\partial t} + \frac{\rho_b}{\theta} \frac{\partial S_i}{\partial t} + v \frac{\partial C_i}{\partial r} = \alpha_L |v| \frac{\partial^2 C_i}{\partial r^2} - \sum_{j=1}^{N-1} [\mathcal{H}(t - t_j^*) - \mathcal{H}(t - t_{j+1}^*)] k_j C_i^{n_j} \pm F_i \quad (2)$$

Here C_i and S_i are the aqueous and sorbed phase concentrations respectively and F_i denotes source/sink terms specified using the user-defined reactions module (described later). The second term on the right hand side of equation (2) allows the user to describe the data using a series of piece-wise linear or non-linear approximations (see Figure 4) and can be thought of as a special case of the more general user-defined module F_i . In (2), \mathcal{H} denotes the Heaviside step function, k_j and n_j are the reaction constants and orders respectively, N denotes the total number of "control points" - times (t_1^* , t_2^* etc.) at which the slope changes in Figure 4 or N (minus one) is the total number of reaction constants used to describe the data and t_j^* are the times at which there is a change in the reaction constants or orders or both (see Fig. 4). Since the Heaviside step function \mathcal{H} operating on a function $f(x)$ turns the function "on" or "off" according to the following property:

$$[\mathcal{H}(t - t_j^*) - \mathcal{H}(t - t_{j+1}^*)] f(x) = \begin{cases} 0 & \text{if } t \leq t_j^* \\ f(x) & \text{if } t_j^* < t < t_{j+1}^* \\ 0 & \text{if } t > t_{j+1}^* \end{cases}$$

the second term on the right hand side of equation (2) can be written as shown below:

$$\begin{aligned} \sum_{j=1}^{N-1} [\mathcal{H}(t - t_j^*) - \mathcal{H}(t - t_{j+1}^*)] k_j C_i^{n_j} = & \left\{ \begin{array}{ll} k_1 C_i^{n_1} & t_1^* \leq t < t_2^* \\ 0 & \text{otherwise} \end{array} \right\} + \left\{ \begin{array}{ll} k_2 C_i^{n_2} & t_2^* \leq t < t_3^* \\ 0 & \text{otherwise} \end{array} \right\} \\ & + \dots + \left\{ \begin{array}{ll} k_{N-1} C_i^{n_{N-1}} & t_{N-1}^* \leq t < t_N^* \\ 0 & \text{otherwise} \end{array} \right\} \end{aligned}$$

The pore water velocity near the well is given by:

$$v = \frac{Q}{2\pi b \theta r} \quad (3)$$

In equation (3), the regional groundwater velocity is assumed to have negligible effect compared to the effects of imposed pumping, a reasonable assumption close to the test well. Before solving equation (2) for the reactive component, we need to invoke a sorption model. PPTEST allows five different sorption models as shown below. In what follows, we focus on equation (2) and drop the suffix i with the understanding that $C = C_i$ and $S = S_i$.

$$\text{Linear equilibrium sorption : } S = K_d C \quad (4)$$

$$\text{Freundlich isotherm : } S = a C^{\frac{1}{m}} \quad (5)$$

$$\text{Langmuir isotherm : } S = \frac{pqC}{1 + pC} \quad (6)$$

$$\text{One - site kinetic sorption : } \frac{\partial S}{\partial t} = \alpha (K_d C - S) \quad (7)$$

$$\text{Two - site kinetic sorption : } \frac{\partial S}{\partial t} = \alpha [(1 - f)K_d C - S] \quad (8)$$

More details about the one-site and two-site sorption models can be found in van Genuchten and Wagenet (1989). For the isotherm-based models, if equations (4), (5) or (6) are differentiated with

respect to time and substituted into equation (2), then we obtain a single equation for the aqueous phase concentration with a retardation factor multiplying the unsteady term as shown below.

$$R_i \frac{\partial C_i}{\partial t} + \frac{A}{r} \frac{\partial C_i}{\partial r} = \alpha_L \left| \frac{A}{r} \right| \frac{\partial^2 C_i}{\partial r^2} - \sum_{j=1}^{N-1} [\mathcal{H}(t - t_j^*) - \mathcal{H}(t - t_{j+1}^*)] k_j C_i^{n_j} \pm F_i$$

where the retardation factor for the i -th species is given by:

$$R_i = 1 + \frac{\rho_b}{\theta} \frac{\partial S_i}{\partial C_i}$$

The retardation factors corresponding to the isotherm models (equations 4-6) appear as shown below.

$$R = 1 + \frac{\rho_b}{\theta} K_d \quad (9)$$

$$R_F = 1 + \frac{\rho_b m C^{m-1}}{\theta} \quad (10)$$

$$R_L = 1 + \frac{\rho_b}{\theta} \left(\frac{pq}{(1 + pC)^2} \right) \quad (11)$$

In equations (9-11) R , R_F and R_L denote the retardation factors corresponding to the linear, Freundlich and Langmuir isotherms respectively. All three isotherms (4-6) are linear for low concentrations; however, the Freundlich and Langmuir isotherms change slope at higher concentrations. The main difference between the Freundlich and Langmuir isotherms is that there is no capacity term (upper limit for sorption) in the Freundlich model while there is an upper limit in the Langmuir model (the term q denotes the capacity term).

Initial conditions for the tracer and the reactive components can be specified either as a zero initial condition or as a background value. Depending on the sorption model used, an additional equation needs to be solved for S . Therefore boundary and initial conditions are shown for both aqueous and sorbed-phase concentrations.

$$C(r, t = 0) = C_b : r > r_w \quad (12)$$

$$S(r, t = 0) = S_b : r > r_w \quad (13)$$

If C_b and S_b are non-zero, they can be specified in lines 193-199 in the program (code should be recompiled). Boundary conditions are different for the injection, rest and extraction phases as shown below (see Figure 1).

Injection of Test Solution:

$$\left(-\alpha_L \frac{\partial C}{\partial r} + C \right)_{r=r_w} = C_0 : 0 < t < t_{inj} \quad (14)$$

$$S(r_w, t) = S_0 : 0 < t < t_{inj} \quad (15)$$

$$\frac{\partial C(r \rightarrow \infty, t)}{\partial r} = 0 \quad (16)$$

$$\frac{\partial S(r \rightarrow \infty, t)}{\partial r} = 0 \quad (17)$$

$$C_{r=r_w} = C_0 : 0 < t < t_{inj} \quad (18)$$

The value of S_0 is calculated inside the code based on the sorption model used. In some cases equation(18) can be used for the boundary condition near the well casing instead of equation (14). Both forms (18 and 14) are implemented in PPTEST although equation (14) is the default condition

used. To use the Robin boundary condition, lines 311 and 320 should be activated (uncommented) and lines 304-309 and 313-318 should be commented. This change requires compilation of the source code.

Injection of Chaser:

The mathematical form of the boundary conditions is identical to those for the injection of a test solution (equations 14 - 18) except that C_0 and S_0 in equations (14) and (15) are replaced with the concentrations for the chaser solution (C_1 and S_1) for each species simulated.

$$\left(-\alpha_L \frac{\partial C}{\partial r} + C\right)_{r=r_w} = C_1 : t_{inj} < t < (t_{inj} + t_{chaser}) \quad (19)$$

$$S(r_w, t) = S_1 : t_{inj} < t < (t_{inj} + t_{chaser}) \quad (20)$$

$$\frac{\partial C(r \rightarrow \infty, t)}{\partial r} = 0 \quad (21)$$

$$\frac{\partial S(r \rightarrow \infty, t)}{\partial r} = 0 \quad (22)$$

$$C_{r=r_w} = C_1 : t_{inj} < t < (t_{inj} + t_{chaser}) \quad (23)$$

Equation (19) can be replaced by equation (23) in the code. Both forms are implemented.

Rest Period:

$$\left(-\alpha_L \frac{\partial C}{\partial r} + C\right)_{r=r_w} = 0 : (t_{inj} + t_{chaser}) < t < (t_{inj} + t_{chaser} + t_{rest}) \quad (24)$$

$$\frac{\partial S(r_w, t)}{\partial r} = 0 : (t_{inj} + t_{chaser}) < t < (t_{inj} + t_{chaser} + t_{rest}) \quad (25)$$

$$\frac{\partial C(r \rightarrow \infty, t)}{\partial r} = 0 \quad (26)$$

$$\frac{\partial S(r \rightarrow \infty, t)}{\partial r} = 0 \quad (27)$$

Extraction:

$$\frac{\partial C(r_w, t)}{\partial r} = 0 : (t_{inj} + t_{chaser} + t_{rest}) < t < t_{max} \quad (28)$$

$$\frac{\partial S(r_w, t)}{\partial r} = 0 : (t_{inj} + t_{chaser} + t_{rest}) < t < t_{max} \quad (29)$$

$$C(r \rightarrow \infty, t) \rightarrow 0 : t > 0 \quad (30)$$

$$S(r \rightarrow \infty, t) \rightarrow 0 : t > 0 \quad (31)$$

In equations (28) and (29) $t_{max} = (t_{inj} + t_{chaser} + t_{rest} + t_{ext})$. While using the kinetic or two-site sorption model, boundary conditions for the sorbed-phase concentration (equations 15, 20) S_0 and S_1 need to be specified. If we assume that the sorbed-phase is locally in equilibrium with the aqueous phase, then the following conditions can be used ($f = 1$ for the one-site kinetic model).

$$S_0 = (1 - f)K_d C_0 \quad (32)$$

$$S_1 = (1 - f)K_d C_1 \quad (33)$$

Equations (32) and (33) appear in lines 222 and 223 in the source code. These lines can be changed to specify different conditions for S_0 and S_1 .

3 Numerical Methods

Computational grids with variable step sizes can be generated using two methods. In the first method, geometrically varying step sizes can be generated. If this method is used, then the user specifies the number of points n and the desired resolution (step size) near the well (Δr_1). r_{\max} can be calculated using equation (34). The geometric ratio λ that produces a desired r_{\max} can be easily calculated by taking the logarithms on both sides of equation (34).

$$r_{\max} = \frac{\Delta r_1 (\lambda^{n_r} - 1)}{(\lambda - 1)} \quad \lambda > 1 \quad (34)$$

In the second method, variable grids can be generated using a nonlinear transform (equations 35 and 36) to obtain refined grid spacing around an arbitrary location r_c .

$$B = \frac{1}{2\gamma} \ln \left[\frac{1 + (e^\gamma - 1) \left(\frac{r_c}{r_{\max}} \right)}{1 + (e^{-\gamma} - 1) \left(\frac{r_c}{r_{\max}} \right)} \right] \quad 0 < \gamma < \infty \quad (35)$$

$$r = r_c \left\{ 1 + \frac{\sinh [\gamma (r' - B)]}{\sinh(\gamma B)} \right\} \quad (36)$$

The first and second derivatives in equations (1) and (2) can be approximated using the following expressions (Phanikumar and Mahajan, 1998) for variable grid spacing (Figure 2).

$$\left. \frac{\partial C}{\partial r} \right|_j = \frac{C_{j+1}}{\beta_j (\Delta r_j + \Delta r_{j-1})} - \frac{\beta_j C_{j-1}}{(\Delta r_j + \Delta r_{j-1})} + \frac{(\Delta r_j - \Delta r_{j-1})}{(\Delta r_j \cdot \Delta r_{j-1})} C_j \quad (37)$$

$$\left. \frac{\partial^2 C}{\partial r^2} \right|_j = \frac{2}{(\Delta r_j + \Delta r_{j-1})} \left[\frac{C_{j+1}}{\Delta r_j} + \frac{C_{j-1}}{\Delta r_{j-1}} \right] - \frac{2C_j}{(\Delta r_j \cdot \Delta r_{j-1})} \quad (38)$$

$$\beta = \left(\frac{\Delta r_j}{\Delta r_{j-1}} \right) \quad (39)$$

In the above equations the suffix j denotes the spatial location (this should not be confused with the summation index j in equation (2)). PPTEST uses (first and second-order accurate) upwind difference schemes to approximate the advection terms. Depending on the direction of velocity, the first-order accurate upwind scheme uses either forward or backward differences to evaluate the first derivative in the advection term. The following equations summarize the second-order accurate upwind scheme. More details of this scheme can be found in Roache (1998).

$$\frac{\partial}{\partial r} (vC) = \frac{v_R C_R - v_L C_L}{\Delta} \quad (40)$$

$$v_R = \frac{1}{2} (v_{j+1} + v_j) \quad (41)$$

$$v_L = \frac{1}{2} (v_j + v_{j-1}) \quad (42)$$

$$\left. \begin{aligned} C_R &= C_j & \text{if } v_R > 0 \\ C_R &= C_{j+1} & \text{if } v_R < 0 \\ C_L &= C_{j-1} & \text{if } v_L > 0 \\ C_L &= C_j & \text{if } v_L < 0 \end{aligned} \right\} \quad (43)$$

The above conditions can be combined into one single equation with four switches ψ_1, ψ_2, ψ_3 and ψ_4 as shown below.

$$\frac{\partial}{\partial x} (vC) = \frac{v_R \psi_1 C_{j+1} + v_R \psi_2 C_j + v_L \psi_3 C_j + v_L \psi_4 C_{j-1}}{\Delta} \quad (44)$$

Using the above approximations, the implicit finite-difference representation of equation (2) appears as shown below.

$$\begin{aligned} \frac{C_j^{\ell+1} - C_j^\ell}{\Delta t} + \frac{\rho_b}{\theta} \left(\frac{\partial S}{\partial t} \right)_j^\ell + \left(\frac{\psi_1 v_R C_{j+1}^{\ell+1} + \psi_2 v_R C_j^{\ell+1} + \psi_3 v_L C_j^{\ell+1} + \psi_4 v_L C_{j-1}^{\ell+1}}{\Delta} \right) \\ = \alpha_L |v|_j^\ell \left\{ \frac{2}{(\Delta r_j + \Delta r_{j-1})} \left[\frac{C_{j+1}^{\ell+1}}{\Delta r_j} - \frac{C_{j-1}^{\ell+1}}{\Delta r_{j-1}} \right] - \frac{2C_j^{\ell+1}}{(\Delta r_j \cdot \Delta r_{j-1})} \right\} - \\ \sum_{i=1}^{N-1} [\mathcal{H}(t - t_i^*) - \mathcal{H}(t - t_{i+1}^*)] k_i (C_j^{\ell+1})^{n_i} \pm F_j^\ell \end{aligned} \quad (45)$$

In the above equation, the superscripts $\ell + 1$ and ℓ denote the new and old time levels respectively. The summation index j in equation (2) is replaced with i in the above equation to avoid confusion with the spatial index j . If the above equation is written once for every grid node, then we obtain a system of algebraic equations in the following form:

$$-\mathbf{E}_j C_{j-1}^{\ell+1} + \mathbf{F}_j C_j^{\ell+1} - \mathbf{G}_j C_{j+1}^{\ell+1} = \mathbf{R}_j \quad (46)$$

If we assume that the two-site sorption model is used, then the coefficients appear as shown below.

$$\mathbf{E}_j = -\frac{v_R \psi_1}{\mathcal{R} \Delta} + \frac{2\alpha_L |v|}{(\Delta r_j + \Delta r_{j-1}) \mathcal{R} \Delta r_j} \quad (47)$$

$$\mathbf{F}_j = \frac{1}{\Delta t} + \frac{v_R \psi_2 + v_L \psi_3}{\mathcal{R} \Delta} + \frac{2\alpha_L |v|}{(\Delta r_j \Delta r_{j-1}) \mathcal{R}} + \frac{k_i}{\mathcal{R}} (C_j^\ell)^{n_i-1} + \delta \frac{\rho_b}{\theta} \alpha (1-f) K_d \quad (48)$$

$$\mathbf{G}_j = -\frac{v_L \psi_4}{\Delta \mathcal{R}} + \frac{2\alpha_L |v|}{(\Delta r_j + \Delta r_{j-1}) \Delta r_{j-1} \mathcal{R}} \quad (49)$$

$$\mathbf{R}_j = \frac{C_j^\ell}{\Delta t} + \delta \frac{\rho_b}{\theta} \alpha S_j^\ell \pm F_j^\ell \quad (50)$$

For the sake of illustration, the above system of equations gives rise to the following tridiagonal matrix system for the case of five grid nodes.

$$\begin{bmatrix} \mathbf{F}_1 & \mathbf{G}_1 & & & \\ \mathbf{E}_2 & \mathbf{F}_2 & \mathbf{G}_2 & & \\ & \mathbf{E}_3 & \mathbf{F}_3 & \mathbf{G}_3 & \\ & & \mathbf{E}_4 & \mathbf{F}_4 & \mathbf{G}_4 \\ & & & \mathbf{E}_5 & \mathbf{F}_5 \end{bmatrix} \begin{bmatrix} C_1^{\ell+1} \\ C_2^{\ell+1} \\ C_3^{\ell+1} \\ C_4^{\ell+1} \\ C_5^{\ell+1} \end{bmatrix} = \begin{bmatrix} \mathbf{R}_1 \\ \mathbf{R}_2 \\ \mathbf{R}_3 \\ \mathbf{R}_4 \\ \mathbf{R}_5 \end{bmatrix} \quad (51)$$

In equations (47-50), Δ is an appropriate step size (Δr_j or Δr_{j-1}) depending on the direction of velocity and \mathcal{R} is a retardation coefficient ($=R, R_F$ or R_L) that depends on the sorption model. If the kinetic or two-site sorption models are used, then $\mathcal{R} = 1$. If the nonlinear equilibrium models are used (i.e., $\mathcal{R} \neq 1$), then $\delta = 0$ in equations (48) and (50). In equation (48), the reaction term was linearized and approximated as:

$$\frac{k_i}{\mathcal{R}}(C_j^{\ell+1})^{n_i} = \frac{k_i}{\mathcal{R}}[(C_j^\ell)^{n_i-1} \cdot (C_j^{\ell+1})] \quad (52)$$

Other approximations are possible (e.g., Crank-Nicholson approximation). The two-site sorption model can be represented as shown below.

$$\frac{S_j^{\ell+1} - S_j^\ell}{\Delta t} = \alpha(1-f)K_d C_j^\ell - S_j^\ell \quad (53)$$

Rearranging the equation in the form (46), we get the following coefficients for the tridiagonal matrix:

$$\mathbf{E}_j = 0 \quad (54)$$

$$\mathbf{F}_j = \frac{1}{\Delta t} + 1 \quad (55)$$

$$\mathbf{G}_j = 0 \quad (56)$$

$$\mathbf{R}_j = \frac{S_j^\ell}{\Delta t} + \alpha(1-f)K_d C_j^\ell \quad (57)$$

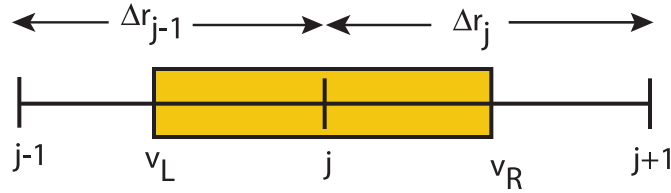


Figure 2: Nomenclature for the finite-difference stencil

Examples of a variable grids generated using equations (35-36) is shown in Figure 3. The step size decreases as we approach r_c and increases away from it. To identify the tuning parameter β that produces the desired resolution at a given location r_c , the following MATLAB script can be used.

```

1 clear; clc;
2 rw=0.104;
3 rmax=50.0;
4 nr=3001;
5 tau=10.0; % this is gamma in eqs. (35-36)
6 rc=0.36-rw;
7 rmax=rmax-rw;
8 %Calculate constants
9 anr = 1.0 + (exp(tau)-1.0)*(rc/rmax);
10 adr = 1.0 + (exp(-tau)-1.0)*(rc/rmax);
11 b = (1.0/(2.0*tau))*log(anr/adr);
12 %First generate a uniform mesh
13 del=1/(nr-1);
14 %del=1.0/(nr-1);
15 rr(1)=0;
16 for i=2:nr
17 %rr(i)=b+(1.0/tau)*asinh((rat-1.0d0)*dsinh(tau*b));
18 rr(i)=rr(i-1)+del;
```

```

19 end
20 %Now generate the variable mesh:
21 r(1)=0;
22 for i=2:nr
23 anum = sinh(tau*(rr(i)-b));
24 aden = sinh(tau*b);
25 r(i)=1.0 + anum/aden;
26 r(i)=rc*r(i);
27 end
28 for i=1:nr
29     r(i)=r(i)+rw;
30 end
31 for i=2:nr
32 dr(i-1)=r(i)-r(i-1);
33 end
34 dr(nr)=dr(nr-1);
35 [r(1) dr(1)]
36 figure(1);
37 i=1:nr;
38 loglog(r,dr,'-');
39 xlabel('Radial distance from well casing, r_{j}');
40 ylabel('Step size, \Delta r_{j}');
41 grid on;
    
```

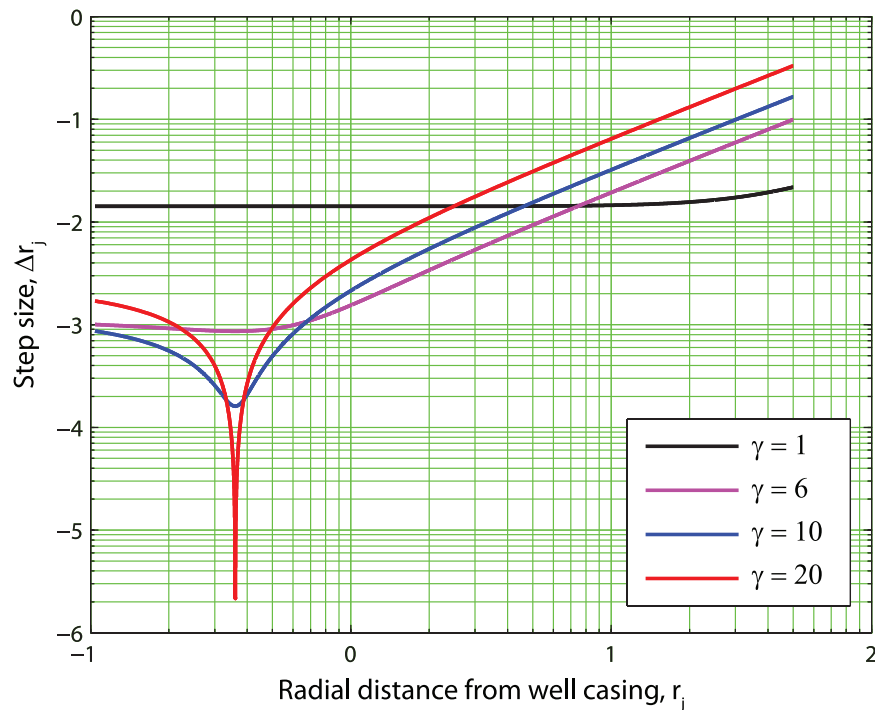


Figure 3: Examples of variable meshes generated by equations (35-36) for different values of the tuning parameter, γ . The resolution around the radial location r_c increases with γ . Note the logarithmic scale for both axes.

The solution of the above system of simultaneous equations (equation 46) can be obtained efficiently using the well-known Thomas algorithm (Roache, 1998).

4 Input Parameters

The number of parameters needed to run PPTEST depends on the sorption and reaction models used. The number and type of parameters will also change depending on whether a uniform or variable mesh is desired. Any consistent system of units can be used as there are no unit conversions inside the code.

Line 1: `rmax, nr, dt`

Line 2: `imesh`

Line 3: `drone,grat`

Line 3: `tau, rdiag`

Line 4: `iadvect`

Line 5: `idiag`

Line 6: `ichasr`

Line 7: `qinj, qext, tinj, trest, text, baquifer,por,alphal,rhob,rw`

Line 8: `qinj2, tinj2`

Line 9: `numsp, nseg`

Line 10: `c0`

Line 11: `korder`

Line 12: `tcrit`

Line 13: `k`

Line 14: `c1`

Line 15: `isorp`

Line 16: `kd`

Line 16: `a,b`

Line 16: `a,b`

Line 16: `alpha,kd`

Line 16: `alpha,frac,kd`

Line 17: `usrxn`

A detailed explanation of these input parameters is given below.

Line 1: `rmax, nr, dt`

1. `rmax`: Distance from the well casing to the outer boundary (r_{\max})
2. `nr`: Number grid points n
3. `dt`: Time step Δt

Line 2: `imesh`

This parameter is used to select a method for generating the computational mesh. If `imesh = 1`, then a uniform mesh is generated and the next line (in which parameters are specified for generating a variable grid) is not required.

Line 3:

If `imesh = 2`, then a variable mesh in which the step size increases geometrically from the well casing will be generated. Parameters `drone,grat` ($\Delta r_1, \lambda$) will be read from line 3.

If `imesh = 3`, then a variable mesh will be generated such that fine grids will be placed near an arbitrary (user-specified) radial location r_c . Parameters γ, r_c (called `tau` and `rdiag` in the code) will be read from line 3. Here γ is a tuning parameter (described earlier).

Line 4: `iadvect`

This parameter specifies the advection scheme. The current version of PPTEST allows the user to use either a first-order or a second-order accurate upwind scheme. Future versions of PPTEST will include other advection schemes.

`iadvect = 1`: First-order accurate upwind scheme `iadvect = 3`: Second-order accurate upwind scheme. `iadvect = 3` is the recommended option for most applications.

Line 5: idiag

Diagnostic node number. Concentration(s) versus time information will be saved to a textfile for the diagnostic node.

Line 6: ichasr

This is a flag (1 = yes, 0 = no) that determines whether the injection period is followed by the injection of a chaser solution. If `ichasr = 1`, then the user is expected to provide details of the chaser injection in separate lines (lines 8 and 14 of the input file).

Line 7: qinj, qext, tinj, trest, text, baquifer,por,alphal,rhob,rw

1. `qinj`: Pumping rate during injection
2. `qext`: Pumping rate during extraction
3. `tinj`: Duration of the injection period
4. `trest`: Duration of the rest period
5. `text`: Duration of the extraction period
6. `baquifer`: Aquifer thickness (b)
7. `por`: Aquifer porosity (θ)
8. `alphal`: Longitudinal dispersivity (α_L)
9. `rhob`: Density (ρ_b)
10. `rw`: Radius of the well casing (r_w)

Line 8: qinj2, tinj2

This line is required only if `ichasr = 1`.

1. `qinj2`: Pumping rate (injection) for the chaser
2. `tinj2`: Duration of the injection period (chaser)

Line 9: numsp, nseg

Number of species to be simulated (`isp = 1` corresponds to a conservative tracer). `nseg` is the number of segments used to describe the data.

Line 10: c0

Injection concentrations for all the species simulated separated by commas (read in the order `c0(i)`, $i = 1, \text{numsp}$)

Line 11: korder

Order of the reaction for the j -th segment, n_j . A total of `nseg` values are expected. PPTTEST allows multiple reaction rates to be used to describe the data as shown in Figure 4. Here, $C_r = c2/c20$ and $C_{tr} = c/c10$ are the normalized concentrations of the reactive component and the tracer respectively. The concentrations are plotted following the simplified method of push-pull test data analysis proposed by Haggerty et al. (1998). In Figure 4, t_1^* , t_2^* and t_3^* are the times `tcrit(1)`, `tcrit(2)` and `tcrit(3)` at which reaction rates (and possibly orders) change. An arbitrary number of reaction rates (and reaction orders) can be simulated although most applications may require zero, one or two reaction rates. Input files for examples 1 through 4 illustrate how to specify these parameters. If the user decides to describe reactions using the more general user-defined reactions module, then it is possible to set `nseg = 0` in line 9, then lines 11, 12 and 13 for `korder`, `tcrit` and `k` are not needed.

Line 12: tcrit

The times t_j^* at which slopes change as shown in Figure 4. A total of `nseg` values are needed. $t_1^* = 0$

Line 13: k

The reaction rates k_j used to describe the different segments as shown in Figure 4. A total of `nseg` values is needed. If there is a lag phase for the reaction, then $k_1 = 0$.

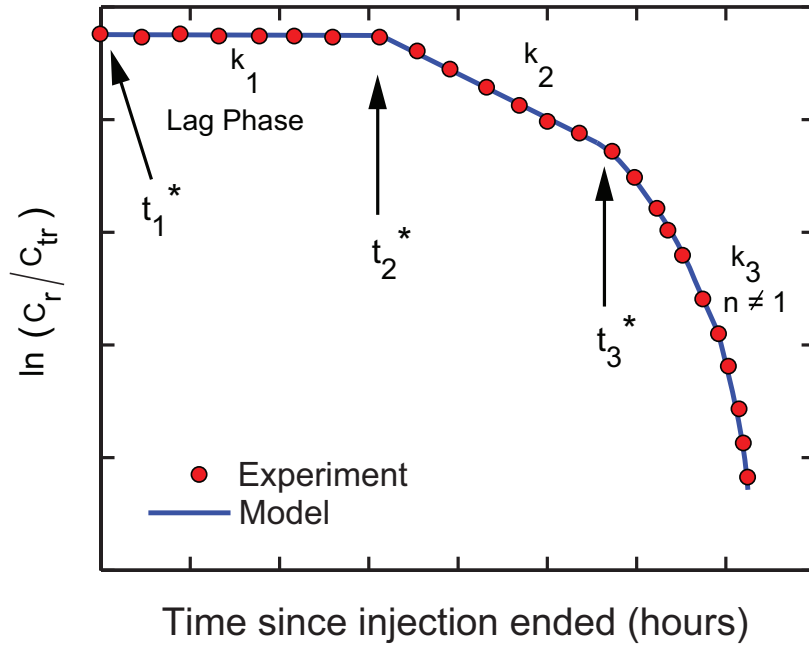


Figure 4: The presence of a lag phase, multiple reaction rates and potentially complex reaction order could complicate the interpretation of rates. PPTEST can simulate situations such as the one shown above.

Line 14: c1

If a second injection period (chaser solution) is simulated, then concentrations of all species during this period need to be specified on this line separated by commas.

Line 15: isorp

This parameter can take values from 0 to 5 and is used to select a sorption model. If `isorp = 0`, then sorption is not simulated and the next line (in which model-specific sorption parameters are specified) is not required.

Line 16:

If `isorp = 1`, then linear equilibrium sorption is simulated. The distribution coefficient `kd` is read from line 16.

If `isorp = 2`, then the Freundlich isotherm is used. Two parameters `a`, `b` (equation 5) are read from line 16.

If `isorp = 3`, then the Langmuir isotherm is simulated. Two parameters `a`, `b` (equation 6) are read from line 16.

If `isorp = 4`, then the one-site kinetic sorption model is used. Two parameters `alpha`, `kd` (equation 7) are read from line 16.

If `isorp = 5`, then the two-site kinetic sorption model is used. For this case, three parameters `alpha`, `frac`, `kd` (equation 8) are read from line 16.

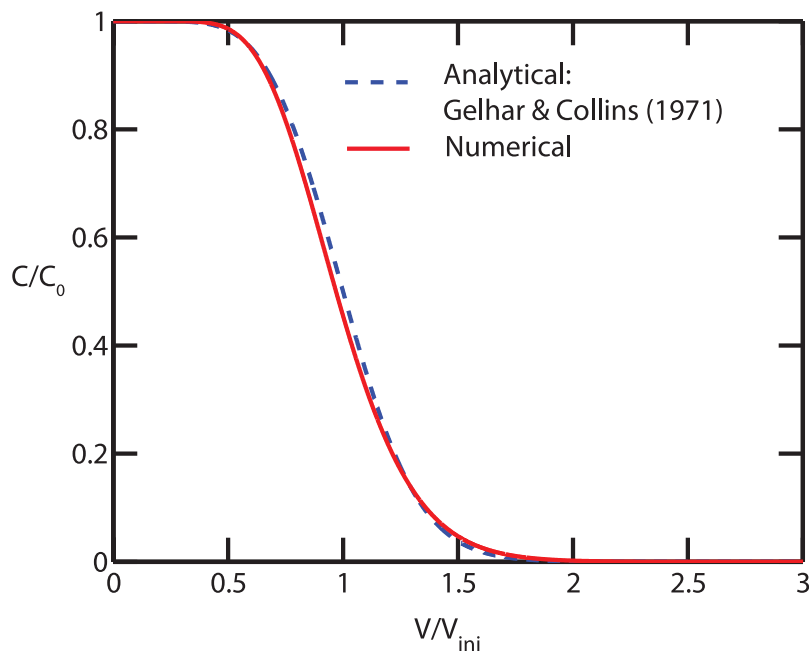


Figure 5: Comparison of numerical results obtained from PPTEST with the approximate analytical solution of Gelhar & Collins (1971). A uniform grid of 1000 points was used to compute the solution. Other parameters are shown in the input file above.

Line 17: usrxn

This is a flag (1 = Yes, 0 = No) to determine if user-defined reactions are simulated. If `usrxn = 1`, then the subroutine `reactions` is called. Example 4 illustrates the use of this subroutine.

5 Examples

In this section we provide several examples to illustrate the application of PPTEST. These examples are discussed in Phanikumar and McGuire (2010).

5.1 Example1: Comparison with the analytical solution of Gelhar & Collins (1971)

Gelhar & Collins (1971) provided the following approximate analytical solution for the radial dispersion problem:

$$\frac{C}{C_0} = \frac{1}{2} \operatorname{erfc} \left[\frac{\left(\frac{V}{V_{inj}} - 1 \right)}{\left\{ \frac{16}{3} \left(\frac{\alpha_L}{r_{max}} \right) \left(2 - \left| 1 - \frac{V}{V_{inj}} \right|^{1/2} \left(1 - \frac{V}{V_{inj}} \right) \right) \right\}^{1/2}} \right] \quad (58)$$

$$r_{max} = \sqrt{\frac{Qt_{inj}}{\pi b \theta R}} \quad (59)$$

where V denotes the cumulative extracted volume ($= |Q_{ext}|t$), $V_{inj} = Q_{inj} \cdot t_{inj}$ and t_{inj} denotes the duration of the injection period. The retardation factor $R = 1$. Any set of parameters can be used to compare with the analytical solution, however, an upper limit for the applicability of the above equation is $\epsilon \ll 0.01$ where $\epsilon = \alpha_L / 2r_{max}$. A numerical solution was computed using PPTEST for the following set of parameters. The analytical solution is computed using the MATLAB script.

```

1 10.0, 1001, 0.1
2 1
3 3
4 2
5 0
6 2.587, 2.282, 94.32, 0.0, 405.6, 8.0, 0.38, 0.064, 1.7, 0.052
7 1, 0
8 1.0
9 0
10 0

```

MATLAB script to compare the numerical solution with the analytical solution of Gelhar & Collins (1971):

```

1 %-----parameters-----
2 qinj = 2.587; qext=2.282;b=8.0; por=0.38;
3 rw= 0.052;tinj=94.32;text=405.6; trest=0.00;alphan=0.064;
4 %-----
5 vinj=qinj*tinj;
6 rmax = sqrt(vinj/(pi*b*por));
7 epsilon = alphan/(2.0*rmax)
8
9 load out1.dat;
10 time=out1(:,2);
11 nt=length(time);
12 Cnum=out1(:,3);
13 %-----Compute Approximate Analytical Solution-----
14 tt = time; % this is extr_vol_by_inj_vol in PPTEST
15 term1 = (tt-1.0);
16 term2 = (abs(1.0-tt)).^0.5;
17 term2 = term2.*(1.0-tt);
18 term2 = (2.0-term2);
19 term2 = ((16.0/3.0)*(alphan/rmax)).*term2;
20 term2 = term2.^0.5;
21 !del a1.dat
22 C = 0.5*erfc(term1./term2);
23 out = [tt Cnum C];
24 save -ascii a1.dat out;
25 plot(tt,C,'b--');
26 hold on;
27 plot (tt,Cnum,'r-');
28 legend('Analytical (Gelhar & Collins (1971))','Numerical');legend boxoff;
29 xlabel('V/V_{inj}','fontweight','bold','fontsize',12); ylabel('C/C0','rotation',0);
30 set(gca,'xlim', [0 3]);
31 %-----

```

5.2 Example 2: Estimation of dispersion and sorption parameters

Pickens et al. (1981) conducted push-pull tests in a sandy aquifer to understand the sorption of ^{85}Sr using ^{131}I as a non-reactive tracer. Tracer breakthrough curves were obtained at several radial distances ($r = 0.36, 0.66$ and 2.06) and depths using multi-level sampling devices during the injection period. During the extraction period samples were collected from the well discharge line. Schroth et al. (2001) compared the results of a simplified method of analysis with the data of Pickens et al. (1981) and determined $\alpha_L = 6.4$ cm and $K_d = 2.33$ ml/g. The input file used for the comparison is

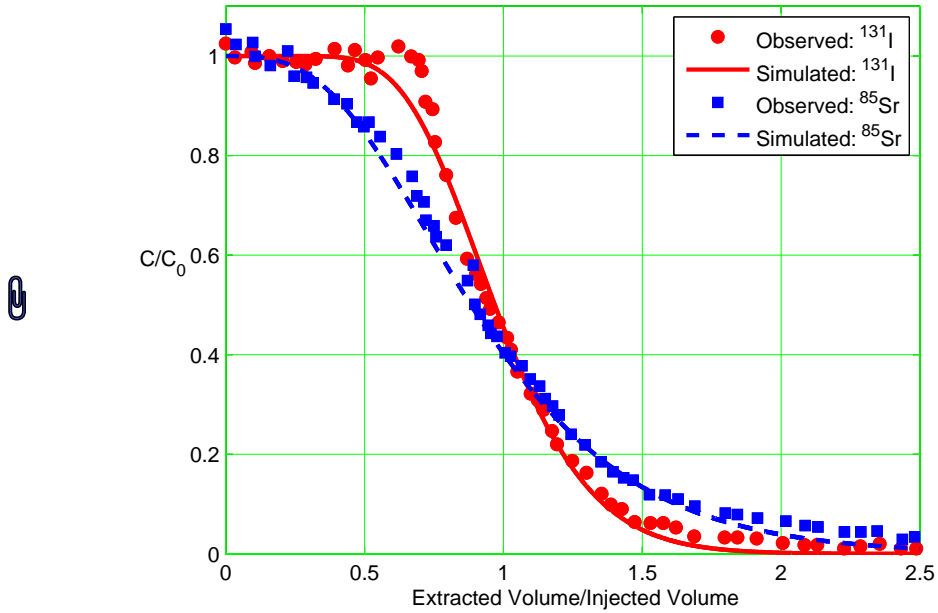


Figure 6: Comparison of numerical results obtained from PPTEST with the observations of Pickens et al. (2002). A variable grid of 5000 points with fine grids clustered around $r_c = 0.36$, a time step of 0.1 and $r_{max} = 10$ were used. Grid clustering was accomplished using a tuning parameter $\beta = 15$. ^{85}Sr was described using the linear equilibrium sorption model ($isorp = 1$).

shown below and uses the same values of α_L and K_d . Comparisons between the observed data and the numerical solutions are shown in Figure 6.

```

1 | 10.0, 5001, 0.1
2 | 3
3 | 15.0, 0.36
4 | 3
5 | 2
6 | 0
7 | 2.587, 2.282, 94.32, 0.0, 405.6, 8.0, 0.38, 0.064, 1.7, 0.052
8 | 2, 3
9 | 1.0, 1.0
10 | 1.0,1.0, 1.0
11 | 0.0, 7000.0, 7000.0
12 | 0.0, 0.0, 0.0
13 | 0,1
14 | 2.33
15 | 0
    
```

The output generated using the above parameters is shown below (out1.dat and out2.dat).

```

out1.dat:
0.08000    0.00075    0.10000E+0001
0.18000    0.00168    0.10000E+0001
0.28000    0.00262    0.10000E+0001
0.38000    0.00355    0.10000E+0001
0.48000    0.00449    0.10000E+0001
0.58000    0.00542    0.10000E+0001
    
```

0.68000	0.00636	0.10000E+0001
0.78000	0.00729	0.10000E+0001
0.88000	0.00823	0.10000E+0001
0.98000	0.00917	0.10000E+0001
1.08000	0.01010	0.10000E+0001
1.18000	0.01104	0.10000E+0001
1.28000	0.01197	0.10000E+0001
1.38000	0.01291	0.10000E+0001
1.48000	0.01384	0.10000E+0001
1.58000	0.01478	0.10000E+0001
1.68000	0.01571	0.10000E+0001
1.78000	0.01665	0.10000E+0001
1.88000	0.01758	0.10000E+0001
1.98000	0.01852	0.10000E+0001
2.08000	0.01945	0.10000E+0001
2.18000	0.02039	0.10000E+0001
----	----	----
----	----	----
79.08000	0.73957	0.83199E+0000
79.18000	0.74051	0.83091E+0000
79.28000	0.74145	0.82983E+0000
79.38000	0.74238	0.82875E+0000
79.48000	0.74332	0.82766E+0000
79.58000	0.74425	0.82657E+0000
79.68000	0.74519	0.82548E+0000
79.78000	0.74612	0.82438E+0000
79.88000	0.74706	0.82328E+0000
79.98000	0.74799	0.82217E+0000
80.08000	0.74893	0.82106E+0000
80.18000	0.74986	0.81995E+0000
80.28000	0.75080	0.81884E+0000
80.38000	0.75173	0.81772E+0000
80.48000	0.75267	0.81660E+0000
80.58000	0.75360	0.81548E+0000
80.68000	0.75454	0.81435E+0000
80.78000	0.75547	0.81322E+0000
80.88000	0.75641	0.81208E+0000
80.98000	0.75734	0.81095E+0000
81.08000	0.75828	0.80981E+0000
81.18000	0.75921	0.80866E+0000
81.28000	0.76015	0.80752E+0000
81.38000	0.76108	0.80637E+0000
81.48000	0.76202	0.80521E+0000
81.58000	0.76296	0.80406E+0000
81.68000	0.76389	0.80290E+0000
81.78000	0.76483	0.80174E+0000
81.88000	0.76576	0.80057E+0000
----	----	----
----	----	----

403.78000	3.77625	0.95051E-0008
403.88000	3.77718	0.94429E-0008
403.98000	3.77812	0.93811E-0008
404.08000	3.77905	0.93198E-0008
404.18000	3.77999	0.92588E-0008
404.28000	3.78092	0.91982E-0008
404.38000	3.78186	0.91380E-0008
404.48000	3.78279	0.90782E-0008
404.58000	3.78373	0.90188E-0008
404.68000	3.78466	0.89598E-0008
404.78000	3.78560	0.89012E-0008
404.88000	3.78653	0.88429E-0008
404.98000	3.78747	0.87851E-0008
405.08000	3.78840	0.87276E-0008
405.18000	3.78934	0.86705E-0008
405.28000	3.79027	0.86138E-0008
405.38000	3.79121	0.85574E-0008
405.48000	3.79214	0.85015E-0008
405.58000	3.79308	0.84459E-0008
405.68000	3.79401	0.83906E-0008

out2.dat :

0.08000	0.00075	0.10000E+0001
0.18000	0.00168	0.10000E+0001
0.28000	0.00262	0.10000E+0001
0.38000	0.00355	0.10000E+0001
0.48000	0.00449	0.10000E+0001
0.58000	0.00542	0.10000E+0001
0.68000	0.00636	0.10000E+0001
0.78000	0.00729	0.10000E+0001
0.88000	0.00823	0.10000E+0001
0.98000	0.00917	0.10000E+0001
1.08000	0.01010	0.10000E+0001
1.18000	0.01104	0.99999E+0000
----	----	----
----	----	----
70.58000	0.66008	0.74186E+0000
70.68000	0.66102	0.74101E+0000
70.78000	0.66195	0.74016E+0000
70.88000	0.66289	0.73931E+0000
70.98000	0.66382	0.73846E+0000
71.08000	0.66476	0.73761E+0000
71.18000	0.66569	0.73676E+0000
71.28000	0.66663	0.73591E+0000
71.38000	0.66756	0.73506E+0000
71.48000	0.66850	0.73420E+0000
71.58000	0.66943	0.73335E+0000
71.68000	0.67037	0.73249E+0000
71.78000	0.67130	0.73164E+0000
71.88000	0.67224	0.73078E+0000



71.98000	0.67317	0.72993E+0000
72.08000	0.67411	0.72907E+0000
72.18000	0.67504	0.72821E+0000
72.28000	0.67598	0.72736E+0000
72.38000	0.67691	0.72650E+0000
72.48000	0.67785	0.72564E+0000
72.58000	0.67879	0.72478E+0000
72.68000	0.67972	0.72392E+0000
72.78000	0.68066	0.72306E+0000
72.88000	0.68159	0.72220E+0000
72.98000	0.68253	0.72134E+0000
73.08000	0.68346	0.72048E+0000
----	----	----
----	----	----
404.68000	3.78466	0.44595E-0003
404.78000	3.78560	0.44491E-0003
404.88000	3.78653	0.44387E-0003
404.98000	3.78747	0.44284E-0003
405.08000	3.78840	0.44180E-0003
405.18000	3.78934	0.44078E-0003
405.28000	3.79027	0.43975E-0003
405.38000	3.79121	0.43872E-0003
405.48000	3.79214	0.43770E-0003
405.58000	3.79308	0.43668E-0003
405.68000	3.79401	0.43566E-0003

5.3 Example3: Estimation of reaction rates

McGuire, et al. (2002) conducted a series of modified push-pull tests at the former Wurtsmith Air Force Base in Michigan. Their study was aimed at quantifying the rates of biogeochemical reactions when recharge water comes in contact with a reduced aquifer. Details of the test conditions are described in McGuire, et al. (2002). Here we use PPTEST to describe the data using a uniform grid of 500 points. Sorption is not simulated for the reactive component (sulfate). The input file is shown below and results are shown in Figure 5.



```

1 | 10.0, 501, 0.01
2 | 1
3 | 3
4 | 8
5 | 1
6 | 0.0333, 0.011, 0.6, 0.0333, 3.6, 0.1, 0.33, 0.001, 1.7, 0.0125
7 | 0.0255,0.067
8 | 2, 3
9 | 100.0, 20.0
10 | 1.0,1.0, 1.0
11 | 0.0, 1.0, 2.5
12 | 0.0, 0.25,1.5
13 | 10.0, 2.0
14 | 0,0
15 | 0
    
```

The output (written to the files `out1.dat` and `out2.dat`) generated by running PPTEST with the above input is shown below. The first column is time since injection ended, the second column is the ratio of volume extracted to the volume injected and the third column is the concentration. If an additional equation is solved for the sorbed-phase concentration, then a fourth column for S appears in the output file(s).

out1.dat:

0.00300	0.00152	0.81999E+0002
0.01300	0.00659	0.81999E+0002
0.02300	0.01167	0.81999E+0002
0.03300	0.01674	0.81999E+0002
0.04300	0.02181	0.83525E+0002
0.05300	0.02688	0.84917E+0002
0.06300	0.03195	0.86188E+0002
0.07300	0.03702	0.87348E+0002
0.08300	0.04210	0.88408E+0002
0.09300	0.04717	0.89375E+0002
0.10300	0.05224	0.90260E+0002
0.11300	0.05731	0.91068E+0002
0.12300	0.06238	0.91806E+0002
0.13300	0.06746	0.92481E+0002
0.14300	0.07253	0.93097E+0002
0.15300	0.07760	0.93661E+0002
0.16300	0.08267	0.94176E+0002
0.17300	0.08774	0.94647E+0002
0.18300	0.09281	0.95077E+0002
0.19300	0.09789	0.95469E+0002
0.20300	0.10296	0.95827E+0002
----	----	----
----	----	----
3.57300	1.81216	0.16269E+0001
3.58300	1.81723	0.15941E+0001
3.59300	1.82230	0.15619E+0001
3.60300	1.82737	0.15304E+0001
3.61300	1.83245	0.14994E+0001
3.62300	1.83752	0.14691E+0001
3.63300	1.84259	0.14393E+0001
3.64300	1.84766	0.14101E+0001

out2.dat:

0.00300	0.00152	0.16400E+0002
0.01300	0.00659	0.16400E+0002
0.02300	0.01167	0.16400E+0002
0.03300	0.01674	0.16400E+0002
0.04300	0.02181	0.16705E+0002
0.05300	0.02688	0.16983E+0002
0.06300	0.03195	0.17238E+0002
0.07300	0.03702	0.17470E+0002
0.08300	0.04210	0.17682E+0002
0.09300	0.04717	0.17875E+0002

0.10300	0.05224	0.18052E+0002
0.11300	0.05731	0.18214E+0002
0.12300	0.06238	0.18361E+0002
0.13300	0.06746	0.18496E+0002
0.14300	0.07253	0.18619E+0002
0.15300	0.07760	0.18732E+0002
0.16300	0.08267	0.18835E+0002
0.17300	0.08774	0.18929E+0002
0.18300	0.09281	0.19015E+0002
0.19300	0.09789	0.19094E+0002
----	----	----
----	----	----
3.52300	1.78680	0.63321E-0001
3.53300	1.79187	0.61154E-0001
3.54300	1.79694	0.59059E-0001
3.55300	1.80201	0.57035E-0001
3.56300	1.80709	0.55079E-0001
3.57300	1.81216	0.53190E-0001
3.58300	1.81723	0.51364E-0001
3.59300	1.82230	0.49600E-0001
3.60300	1.82737	0.47896E-0001
3.61300	1.83245	0.46250E-0001
3.62300	1.83752	0.44659E-0001
3.63300	1.84259	0.43122E-0001
3.64300	1.84766	0.41638E-0001

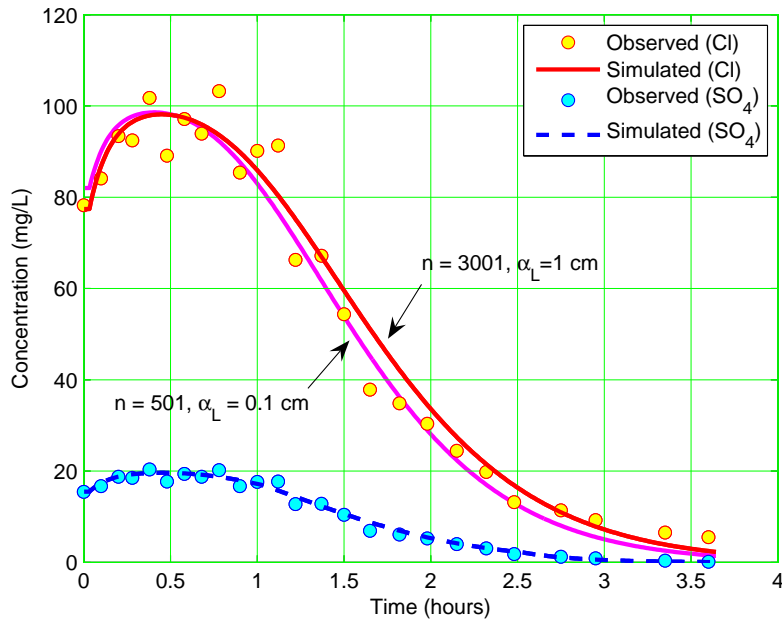


Figure 7: Comparison of numerical results obtained from PPTEST with the observations of McGuire et al. (2002). Two grid sizes (with 501 and 3001 grid points) were used to obtain the results. Both grids used an r_{\max} of 10 and a porosity of 0.33. The diagnostic node was placed at a radial distance of 0.15 ($\text{idiag}=8$ for $n=501$ and $\text{idiag}=43$ for $n=3001$). Both simulations used a time step $\Delta t = 0.01$ hours.

5.4 Example 4: Monod Kinetics / User-Defined Reactions

To illustrate the application of the user-defined reactions module, we consider the following set of coupled differential equations with source terms. Equation (61) has a degradation term based on Monod kinetics (k is a degradation rate and k_S denotes a half-saturation coefficient). The analytical solutions for this coupled system of equations were obtained using the method of manufactured solutions described in Roache (2009). Maple 13 (Waterloo Maple, 2009) was used to simplify the source terms S_1 and S_2 and to generate the Fortran code in the subroutine `reactions`.

$$\frac{\partial C_1}{\partial t} + \frac{A}{r} \frac{\partial C_1}{\partial r} = \alpha_L \left| \frac{A}{r} \right| \frac{\partial^2 C_1}{\partial r^2} + S_1 \quad (60)$$

$$\frac{\partial C_2}{\partial t} + \frac{A}{r} \frac{\partial C_2}{\partial r} = \alpha_L \left| \frac{A}{r} \right| \frac{\partial^2 C_2}{\partial r^2} - k \left(\frac{C_1}{k_S + C_1} \right) C_2 + S_2 \quad (61)$$

$$S_1 = -\frac{1}{4r^{3/2}} e^{-\sqrt{r}t} \left(4r^2 + 2At + \alpha \left| \frac{A}{r} \right| t + \alpha \left| \frac{A}{r} \right| t^2 \sqrt{r} \right) \quad (62)$$

$$S_2 = -\frac{e^{-\frac{\sqrt{r}t}{2\pi}}}{16\pi^2 r^{3/2} (k_S + e^{-\sqrt{r}t})} \left[\left(8r^2\pi + 4At\pi + 2\alpha \left| \frac{A}{r} \right| t\pi + \alpha \left| \frac{A}{r} \right| t^2 \sqrt{r} \right) (k_S + e^{-\sqrt{r}t}) - 16ke^{-\sqrt{r}t}\pi^2 r^{3/2} \right] \quad (63)$$

The above system of equations has the following analytical solutions:

$$C_1 = e^{-t\sqrt{r}} \quad (64)$$

$$C_2 = e^{-\frac{t\sqrt{r}}{2\pi}} \quad (65)$$

The input file for simulating the above system is shown below. The source terms and the Monod term are included in the subroutine `reactions`. The last line of the input file sets the flag `usrxns = 1` which invokes the user-defined reaction module.

```

1 | 2.0, 1001, 0.01
2 | 1
3 | 3
4 | 2
5 | 0
6 | 2.0,0.1, 1.0,0.0,10.0,0.1, 0.33, 0.01, 1.7, 0.01
7 | 2, 1
8 | 1.0, 1.0
9 | 0
10 | 0.0
11 | 0.0
12 | 0, 0
13 | 1

```

Comparisons shown in Figure 8 indicate that PPTTEST has the ability to simulate coupled system of equations such as (60-61) based on Monod or Michelis-Menten kinetics. The user-defined module can also be used to simulate transport based approaches such as the mobile-immobile or residence time distribution (RTD) modeling. Additional examples illustrating the application of PPTTEST will be included in future versions of this document (available at the site: <http://www.egr.msu.edu/~phani/pptest>).

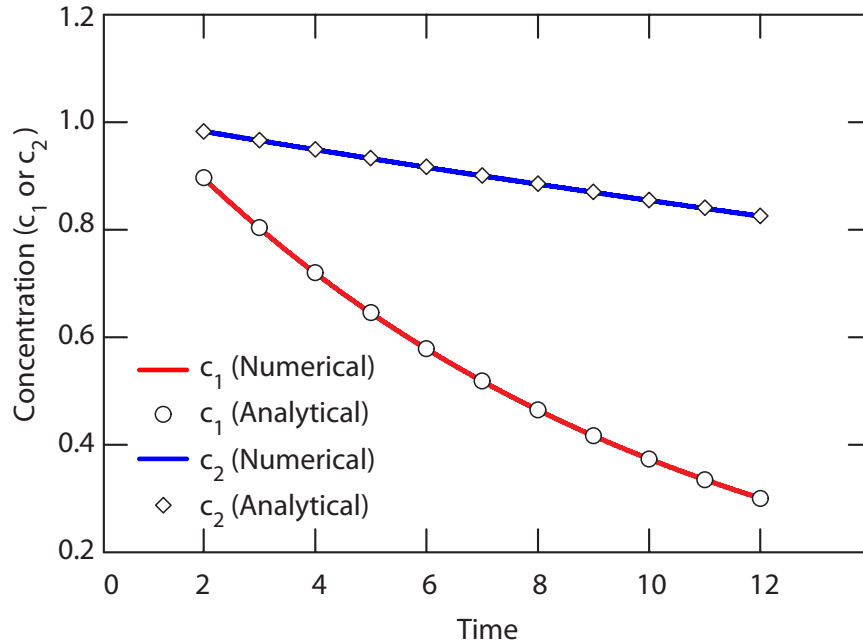


Figure 8: Comparison of numerical results obtained from PPTEST with the analytical solutions for the system of equations (58-62)

6 Program Listing

```

1 program ppt
2 implicit real *8 (a-h,o-z)
3 !There are 5 lines similar to the one below in the code. Change the numbers here
4 !if you want to increase the grid size (mr), the number of components simulated (ns)
5 !or the number of piecewise-linear segments used to describe the data
6 ! on a plot of log(c_r/c_tr) versus time since injection. Here c_r is the reactive component
7 !and c_tr is the tracer.
8 PARAMETER (mr=10001,ncomp=10, ns=10)
9 !-----
10 !Written by Mantha Phanikumar, Michigan State Univervity (phani@msu.edu)
11 !DISCLAIMER: This software is distributed in the hope that it will be useful.
12 !Author makes no warranty that the software will meet your requirements
13 !and shall not be held liable for any damages resulting from the correct
14 !or incorrect use of the software. The software and its documentation
15 !could include technical or other mistakes, inaccuracies or typographical
16 !errors. Efforts will be made to correct any errors in future versions.
17 !-----
18 ! use any consistent system of units. no unit conversions in code.

19 real *8 kd, k, korder, kprime, canaly(10)
20 integer isorp, nr, i, iadvect, userxn
21 character *80 filnam

22 common/param1/r(mr),dr(mr),c(ncomp,mr),cold(ncomp,mr),z1(mr),z2(mr),aa(mr),bb(mr),cc(mr),&
23 dd(mr),s(ncomp,mr),sold(ncomp,mr),v(mr),vt(mr),d(mr),dtracer(mr),qinj,qext,&
24 tinj,trest,text,baquifer,por,rhob,rw,rmax,dt,qinj2,tinj2,tmax,&
25 kd(ncomp),a(ncomp),b(ncomp),alpha(ncomp),frac(ncomp),drone,grat,tau,rdiag,alphal,&
26 c0(ncomp),c1(ncomp),kprime,k(ns),korder(ns),tcrit(ns),time,time_since_inj,&

```

```

27 vol_extr_by_vol_inj, Ak, cone, am, q1r, qmr, pi, flag, flag2, q, retard
28 common/param2/nr,idiag, ichasr,isorp(ncomp),imesh,numsp, nseg,i,isp,itr,iadvect,nt,l1,lm,userxn
29 common/userdefparam/uk(10)
30 write(*,*) 'what is the name of the file (input parameters will be read from this file)?'
31 read(*,*) filnam
32 open(unit=1,file=filnam)
33 !-----
34 ! Read mesh-related parameters
35 !-----
36 drone=0.0d0; grat=0.0d0; tau=0.0d0; rdiag=0.0d0;
37 do i=1,ncomp
38 isorp(i)=0
39 enddo

40 read (1,*) rmax, nr, dt
41 read (1,*) imesh
42 if (imesh.eq.2) read (1,*) drone,grat
43 if (imesh.eq.3) read (1,*) tau, rdiag
44 read (1,*) iadvect
45 read (1,*) idiag
46 read (1,*) ichasr
47 !-----
48 !Injection, extraction and other parametrs
49 !-----
50 read (1,*) qinj, qext, tinj, trest, text, baquifer,por,alphan,rhob,rw
51 if(ichasr.eq.1) then
52 read (1,*) qinj2, tinj2
53 else
54 qinj2=0.0d0
55 tinj2=0.0d0
56 endif
57 !-----
58 !Number of species, number of piece-wise linear segments
59 read (1,*) numsp, nseg
60 !-----
61 if (numsp.eq.0) then
62 write(*,*) 'Number of species is zero! STOP.'
63 STOP
64 endif

65 read (1,*) (c0(i), i=1,numsp)
66 if (nseg.gt.0) then
67 read (1,*) (korder(i),i=1,nseg)
68 read (1,*) (tcrit(i), i=1,nseg)
69 read (1,*) (k(i), i=1,nseg)
70 endif

71 !-----
72 !input chaser concentration
73 if(ichasr.eq.1) then
74 read (1,*) (c1(i), i=1,numsp)
75 endif
76 !-----
77 a=0.0d0; b=0.0d0; alpha=0.0d0; frac=0.0d0;
78 read (1,*) (isorp(i),i=1,numsp)

```

```

79 do i = 1,numsp
80 if (isorp(i).eq.1) read (1,*) kd(i)           !linear sorption
81 if (isorp(i).eq.2) read (1,*) a(i),b(i)     !freundlich isotherm
82 if (isorp(i).eq.3) read (1,*) a(i),b(i)     !langmuir isotherm
83 if (isorp(i).eq.4) read (1,*) alpha(i),kd(i) !one-site kinetic sorption
84 if (isorp(i).eq.5) read (1,*) alpha(i),frac(i),kd(i) !two-site kinetic sorption
85 enddo

86 read (1,*) userxn

87 close(unit=1)

88 !isp is the species number (goes from 1 to nsp). It will be updated inside pptest
89 call pptest

90 stop
91 end
92 !
93 subroutine pptest
94 implicit real *8 (a-h,o-z)
95 PARAMETER (mr=10001,ncomp=10, ns=10)
96 integer isorp, nr, i, iadvect, userxn
97 real *8 k,korder,tcrit, kd, kprime, canaly(10)
98 character *80 filnam
99 character *3 cisp
100 character *20 fname

101 common/param1/ r(mr),dr(mr),c(ncomp,mr),cold(ncomp,mr),z1(mr),z2(mr), aa(mr),bb(mr),cc(mr),&
102 dd(mr), s(ncomp,mr),sold(ncomp,mr),v(mr),vt(mr),d(mr),dtracer(mr),qinj,qext,&
103 tinj, trest, text, baquifer, por, rhob, rw,rmax, dt, qinj2, tinj2,tmax,&
104 kd(ncomp), a(ncomp),b(ncomp), alpha(ncomp), frac(ncomp), drone,grat,tau, rdiag, alphas,&
105 c0(ncomp), c1(ncomp),kprime, k(ns),korder(ns), tcrit(ns), time, time_since_inj,&
106 vol_extr_by_vol_inj, Ak, cone, am, q1r, qmr, pi, flag, flag2, q, retard
107 common/param2/nr,idiag, ichasr,isorp(ncomp),imesh,numsp,nseg,i,isp, itr,iadvect,nt,l1,lm,userxn
108 common/userdefparam/uk(10)
109 !c-----
110 !the times tcrit(1) and tcrit(2) where slopes change (on a plot of
111 ![log(c_r/c_tr) vs t]) are counted from the beginning of injection.
112 pi = 4.0d0*datan(1.0d0)

113 !Generate a log file with all input data
114 open(unit=1, file = 'log.txt')

115 write (1,*) 'rmax, nr, dt:', rmax, nr, dt
116 write (1,*) 'imesh:', imesh
117 if (imesh.eq.2) write (1,*) 'drone,grat:', drone,grat
118 if (imesh.eq.3) write (1,*) 'tau, rdiag:', tau, rdiag
119 write (1,*) 'iadvect:', iadvect
120 write (1,*) 'idiag:', idiag
121 write (1,*) 'ichasr:', ichasr
122 write (1,*) 'qinj, qext, tinj, trest, text, baquifer,por,alphal,rhob,rw:'
123 write (1,*) qinj, qext, tinj, trest, text, baquifer,por,alphal,rhob,rw
124 if(ichasr.eq.1) then
125 write (1,*) 'qinj2, tinj2:'
126 write (1,*) qinj2, tinj2
127 endif

```

```

128 write (1,*) 'numsp, nseg:'
129 write (1,*) numsp, nseg
130 !-----
131 write(1,*) 'c0 values for all species:'
132 write(1,*) (c0(i), i=1,numsp)
133 if (nseg.gt.0) then
134 write (1,*) 'korder for all piece-wise linear segments:'
135 write (1,*) (korder(i),i=1,nseg)
136 write (1,*) 'times (tcrit) at which slopes change for all segments (tcrit(1) = 0):'
137 write (1,*) (tcrit(i), i=1,nseg)
138 write(1,*) 'reaction ks for all segments:'
139 write (1,*) (k(i), i=1,nseg)
140 endif
141 !input chaser concentration
142 if(ichasr.eq.1) then
143 write(1,*) 'chaser concentration for all species:'
144 write (1,*) (c1(i), i=1,numsp)
145 endif
146 write (1,*) 'Sorption-related parameters:'
147 write(1,*) 'isorp for all species:'
148 write (1,*) (isorp(i),i=1,ncomp)
149 do i=1,numsp
150 if (isorp(i).eq.1)then
151 write (1,*) 'isp, isorp(isp), kd(isp): ', i,isorp(i),kd(i)
152 endif
153 if (isorp(i).eq.2) then
154 write (1,*) 'isp, isorp(isp), a(isp), b(isp):', i, isorp(i),a(i),b(i)
155 endif
156 if (isorp(i).eq.3) then
157 write (1,*) 'isp, isorp(isp), a(isp), b(isp):', i, isorp(i),a(i),b(i)
158 endif
159 if (isorp(i).eq.4) then
160 write (1,*) 'isp, isorp(isp), alpha(isp), kd(isp):',i,isorp(i),alpha(i),kd(i)
161 endif
162 if (isorp(i).eq.5) then
163 write (1,*) 'isp, isorp(isp), alpha(isp), frac(isp), kd(isp):',i,isorp(i),alpha(i),frac(i),kd(i)
164 endif
165 enddo
166 close(1)
167 if (rw.eq.0.0) then
168 write(*,*)
169 write(*,*) 'error: radius of well casing is zero (rw = 0). stop'
170 write(*,*)
171 stop
172 endif
173 if (ichasr.eq.0) then
174 qinj2=0.0d0
175 tinj2=0.0d0
176 endif
177 tinjtotal = tinj + tinj2
178 nrr=nr-1
179 iref = 2
180 if (imesh.eq.1) then
181 call umesh(rw,nr,rmax,dr,r)
182 elseif (imesh.eq.2) then
183 call gmesh(rw,drone,grat,nr,dr,r)

```

```

184 elseif (imesh.eq.3) then
185 call vmesh(rw, tau,rmax,rdiag,nr,dr,r)
186 endif
187 open (unit=3, file='mesh.txt')
188 do i=1,nr
189 write(3,*) i,dr(i),r(i)
190 enddo
191 close(3)
192 !c-----initialize variables-----
193 do isp=1,numsp
194 do 16 i=2,nr
195 c(isp,i)=0
196 cold(isp,i)=0
197 s(isp,i)=0.0d0
198 sold(isp,i)=0.0d0
199 16 continue
200 enddo
201 itr=0
202 time = 0.0d0
203 ls = 0
204 tmax = tinjtotal + trest + text
205 nt= (tmax/dt) + 1
206 do j = 1,nseg
207 tcrit(j)=tinjtotal+trest+tcrit(j)
208 enddo

209 1 continue
210 kount=kount+1
211 itr=itr+1
212 time = time + dt
213 time_since_inj = (time-tinjtotal)
214 vol_extr_by_vol_inj = time_since_inj*qext/(qinj*tinj + qinj2*tinj2)

215 do isp = 1,numsp !Solve for all species

216 if ((isorp(isp).eq.0).or.(isorp(isp).ge.4)) retard = 1.0d0
217 if (isorp(isp).eq.1) retard = 1.0d0 + rhob*kd(isp)/por
218 if (isorp(isp).eq. 4) frac(isp) = 1.0d0
219 if (isp.eq.1) retard = 1.0d0

220 if (isorp(isp).ge.4) then
221 retard = 1.0d0
222 s0=(1.0d0-frac(isp))*kd(isp)*c0(isp) ! c0 is conc. for first injection
223 s1=(1.0d0-frac(isp))*kd(isp)*c1(isp) ! c1 is conc. for chaser
224 endif

225 !c-----
226 !c specify boundary-conditions:
227 ! Here c is the aq. phase and s is the sorbed phase

228 if(time.le.tinj) then
229 c(isp,1)=c0(isp)
230 cold(isp,1)=c0(isp)
231 cone = c0(isp)
232 c(isp,nr)=c(isp,nrr) ! neumann conditon
233 cold(isp,nr)=cold(isp,nrr)

```

```

234 if(isorp(isp).ge.4) then
235 s(isp,1)=s0
236 sold(isp,1)=s0
237 s(isp,nr)=s(isp,nrr)
238 sold(isp,nr)=sold(isp,nrr)
239 endif
240 flag=1.0d0
241 q=qinj
242 flag2=1.0d0
243 elseif (((time.gt.tinj).and.(time.lt.tinjtotal)).and.(ichasr.eq.1))) then
244 c(isp,1)=c1(isp)
245 cold(isp,1)=c1(isp)
246 cone = c1(isp)
247 c(isp,nr)=c(isp,nrr)          ! neumann conditon
248 cold(isp,nr)=cold(isp,nrr)
249 if(isorp(isp).ge.4) then
250 s(isp,1)=s1
251 sold(isp,1)=s1
252 s(isp,nr)=s(isp,nrr)
253 sold(isp,nr)=sold(isp,nrr)
254 endif
255 flag=1.0d0
256 q=qinj2
257 flag2=1.0d0
258 elseif ((time.gt.tinjtotal).and.(time.lt.(trest+tinjtotal))) then
259 ss=alpha/dr(1)      ! using mixed boundary conditions  (c(1) = cold(2)*ss/(1+ss))
260 c(isp,1)=c(isp,2)*(ss/(1.0d0+ss))
261 !cold(isp,1)=cold(isp,2)*(ss/(1.0d0+ss))
262 c(isp,nr)=c(isp,nrr)
263 cold(isp,nr)=cold(isp,nrr)
264 if(isorp(isp).ge.4) then
265 s(isp,nr)=s(isp,nrr)
266 !sold(isp,nr)=sold(isp,nrr)
267 endif
268 q=0.0d0  !check
269 flag = 0.0d0
270 flag2=0.0d0
271 elseif ((time.gt.(trest+tinjtotal)).and.(time.le.tmax)) then
272 c(isp,1)=c(isp,2)
273 !cold(isp,1)=cold(isp,2)
274 c(isp,nr)=c(isp,nrr)  ! neumann conditon
275 !cold(isp,nr)=cold(isp,nrr)
276 if(isorp(isp).ge.4) then
277 s(isp,1)=s(isp,2)
278 !sold(isp,1)=sold(isp,2)
279 s(isp,nr)=s(isp,nrr)
280 !sold(isp,nr)=sold(isp,nrr)
281 endif
282 flag = -1.0d0
283 q=qext
284 flag2=1.0d0
285 endif

286 !uncomment these lines for example 4 in the paper
287 !(user-defined reactions with source terms: analytical solution)
288 !flag = 1.0d0;

```

```

289 !flag2=1.0d0;
290 !q=qinj;

291 Ak = flag*q/(2.0d0*pi*baquifer*por)

292 do i=1,nr
293 if (isorp(isp).eq.2) retard = 1.0d0 + rhob*a(isp)*b(isp)*c(isp,i)**(b(isp)-1.0d0)/por
294 if (isorp(isp).eq.3) retard = 1.0d0 + (rhob/por)*(a(isp)*b(isp)/(1.0d0+a(isp)*c(isp,i)**2)
295 vt(i)=Ak/r(i)
296 v(i)=vt(i)/retard
297 dtracer(i)=flag2*alpha*dabs(vt(i))
298 d(i)=flag2*alpha*dabs(v(i))
299 enddo

300 !-----
301 !           boundary conditions
302 !-----

303 if(time.le.tinj) then
304 l1=1
305 cone=c0(isp)
306 lm=2
307 am=0.0d0
308 q1r=c0(isp)
309 qmr=0.0d0 !injection, dirichlet bc
310 ! Third-type BCs can be used at the well during injection by activating the following line:
311 !l1=3; cone=ss;cone2=ss; lm=2; am=0.0d0; q1t=c10;qmt=0.0d0;q1r=c20;qmr=0.0d0
312 elseif (((time.gt.tinj).and.(time.le.tinjtotal)).and.(ichasr.eq.1))) then
313 l1=1;
314 cone=c1(isp);
315 lm=2;
316 am=0.0d0;
317 q1r=c1(isp);
318 qmr=0.0d0 !injection, chaser, dirichlet
319 ! Third-type BCs:
320 !l1=3; cone=ss;cone2=ss; lm=2; am=0.0d0; q1=c110;qm=0.0d0;q1r=c220;qmr=0.0d0
321 elseif ((time.gt.tinjtotal).and.(time.le.(trest+tinjtotal))) then
322 l1=3;
323 cone=ss;
324 lm=2;
325 am=0.0d0;
326 q1r=0.0d0;
327 qmr=0.0d0 !rest:
328 elseif ((time.gt.(trest+tinjtotal)).and.(time.le.tmax)) then
329 l1=2;
330 cone=0.0d0;
331 lm=1;
332 am=0.0d0;
333 q1r=0.0d0;
334 qmr=0.0d0
335 endif

336 !-----
337 !           solve for concentrations:
338 !-----
339 do 882 i=2,nrr

```

```

340 if(nseg.gt.0) then
341 do 8822 j=1,nseg-1
342 if((time.gt.tcrit(j)).and.(time.le.tcrit(j+1))) then
343 kprime = (k(j)/retard)*(c(isp,i)**(korder(j) - 1.0d0)
344 elseif((time.gt.tcrit(nseg))) then
345 kprime=(k(nseg)/retard)*(c(isp,i)**(korder(nseg) - 1.0d0)
346 endif
347 8822 continue
348 else
349 kprime = 0.0d0
350 endif

351 if(isp.eq.1) kprime=0 ! isp=1 is always a tracer

352 !first-order upwind differencing:
353 if(iadvect.eq.1) then
354 call coef1
355 ! elseif (iadvect.eq.2) then
356 ! central differncng - not available in this version
357 ! call coef2
358 elseif (iadvect.eq.3) then
359 call coef3
360 else
361 write(*,*) 'iadvect value out of range! STOP.'
362 STOP
363 endif

364 882 continue

365 call thomas(aa,bb,cc,dd,nr,z1,l1,cone,lm,am,q1r,qmr)
366 do 883 jj=1,nr
367 c(isp,jj)=z1(jj)
368 883 continue

369 !-----
370 ! Depending on the sorption model solve for the sorbed-phase concentration:
371 !-----

372 if (isorp(isp).ge.4) then
373 do i = 2,nrr
374 aa(i)=0.0d0
375 bb(i)=1.0d0 + (1.0d0/dt)
376 cc(i)=0.0d0
377 dd(i)= (1.0d0/dt)*s(isp,i) + alpha(isp)*(1.0d0-frac(isp))*kd(isp)*c(isp,i)
378 enddo
379 if(time.le.tinj) then
380 l1=1; sone=s0; lm=2; am=0.0d0; q1=0.0d0; qm=0.0d0
381 elseif (((time.gt.tinj).and.(time.le.tinjttotal)).and.(ichasr.eq.1))) then
382 l1=1; sone=s1; lm=2; am=0.0d0 ; q1=0.0d0; qm=0.0d0
383 elseif ((time.gt.tinjttotal).and.(time.le.(trest+tinjttotal))) then
384 l1=2; sone=0.0d0;lm=2; am=0.0d0 ; q1=0.0d0; qm=0.0d0
385 elseif ((time.gt.(trest+tinjttotal)).and.(time.le.tmax)) then
386 l1=2; sone=0.0d0; lm=1; am=0.0d0 ; q1=0.0d0; qm=0.0d0
387 endif
388 call thomas (aa,bb,cc,dd,nr,z2,l1,sone,lm,am,q1,qm)
389 do jj=1,nr

```

```

390 s(isp,jj)=z2(jj)
391 enddo
392 endif

393 !=====
394 !      writing the output file
395 !=====
396 !following 2 lines useful to generate the analytical solution in example 4
397 !canaly(1) = dexp(-dsqrt(r(idiag))*time)
398 !canaly(2) = dexp(-dsqrt(r(idiag))*time/pi/0.2D1)

399 ! The following logic creates as many output files as there are number of species
400 !(one file for each species):
401 write(cisp, '(i3)') isp
402 fname = 'out'//trim(adjustl(cisp))//'.dat'

403 open(unit=3,file=fname, access='append')
404 if (time_since_inj.ge.0.0d0) then
405   if (isorp(isp).ge.4) then
406     write(3,11) time_since_inj, vol_extr_by_vol_inj, c(isp,idiag), s(isp,idiag)
407   else
408     write(3,12) time_since_inj, vol_extr_by_vol_inj, c(isp,idiag)
409   endif
410 11  format(f12.5,1x,f12.5, 2x, e15.5e4,2x,e15.5e4)
411 12  format(f12.5,1x,f12.5, 2x, e15.5e4)

412   endif
413 close(unit=3)
414 do 289 i=1,nr
415 cold(isp,i)=c(isp,i)
416 289 continue

417 if (isorp(isp).ge.4) then
418 do 290 i=1,nr
419 sold(isp,i)=s(isp,i)
420 290 continue
421 endif

422 enddo ! this is where the isp loop ends

423 if (itr.ge.nt) then
424 ls = 1
425 itr=0
426 else
427 endif
428 if(ls.eq. 1) goto 2992
429 goto 1

430 2992 stop
431 end
432 !-----
433 subroutine thomas(e,f,g,r,nr,phi,lbc1,valbc1,lbcn,valbcn,q1,qm)
434 implicit real *8 (a-h,o-z)
435 dimension e(nr),f(nr),g(nr),r(nr),a(nr),b(nr),phi(nr)
436 if(lbc1.eq.1) a(1)=0.0d0
437 if(lbc1.eq.1) b(1)=valbc1

```

```

438 if(lbc1.eq.2) a(1)=1.0d0
439 if(lbc1.eq.2) b(1)=-valbc1
440 if(lbc1.eq.3) a(1)=valbc1/(valbc1-1.0d0)
441 if(lbc1.eq.3) b(1)=q1/(1.0d0-valbc1)
442 nrr=nr-1
443 do 1 i=2,nrr
444 dd=1.0d0/(f(i)-g(i)*a(i-1))
445 a(i)=e(i)*dd
446 1 b(i)=(r(i)+g(i)*b(i-1))*dd
447 if(lbcn.eq.1) phi(nr)=valbcn
448 if(lbcn.eq.2) phi(nr)=(b(nrr)+valbcn)/(1.0d0-a(nrr))
449 if(lbcn.eq.3) phi(nr)=(b(mm)+qm/valbcn)/((1.0d0+valbcn)/valbcn-a(nrr))
450 do ii=1,nrr
451 m=nr-ii
452 phi(m)=a(m)*phi(m+1)+b(m)
453 enddo
454 return
455 end
456 !-----
457 subroutine coef1
458 !first-order upwind differencing:
459 implicit real *8 (a-h,o-z)
460 PARAMETER (mr=10001,ncomp=10, ns=10)
461 real *8 k1,k2, kd, kprime, kprime2
462 integer isorp, nr, i, iadvect, userxn

463 common/param1/ r(mr),dr(mr),c(ncomp,mr),cold(ncomp,mr),z1(mr),z2(mr),aa(mr),bb(mr),cc(mr),&
464 dd(mr),s(ncomp,mr),sold(ncomp,mr),v(mr), vt(mr), d(mr),dtracer(mr), qinj, qext,&
465 tinj, trest, text, baquifer, por, rhob, rw,rmax,dt,qinj2,tinj2,tmax,&
466 kd(ncomp), a(ncomp),b(ncomp), alpha(ncomp), frac(ncomp), drone,grat,tau,rdiag,alphan,&
467 c0(ncomp), c1(ncomp),kprime, k(ns),korder(ns),tcrit(ns),time,time_since_inj,&
468 vol_extr_by_vol_inj, Ak, cone, am, q1r, qmr, pi, flag, flag2, q, retard
469 common/param2/nr,idiag, ichasr,isorp(ncomp),imesh,numsp, nseg,i,isp,itr,iadvect,nt,l1,lm,userxn
470 common/userdefparam/uk(10)

471 if (userxn.eq.1) then
472 call reactions(rxn)
473 else
474 rxn = 0.0d0
475 endif

476 if (v(i).ge.0.0d0) then
477 s1=0.0d0
478 s2=1.0d0
479 s3=-1.0d0
480 del = dr(i-1)
481 vv= v(i)
482 vvt=vt(i)
483 else
484 del = dr(i)
485 s1=1.0d0
486 s2=-1.0d0
487 s3=0.0d0
488 vv= v(i)
489 vvt=vt(i)
490 endif

```

```

491 drs=dr(i)+dr(i-1)
492 drp=dr(i)*dr(i-1)

493 !-----
494 !           simultaneous equations
495 !-----
496 if (isp.eq.1) then
497 aa(i)=-vvt*s1/del + 2.0d0*dtracer(i)/(drs*dr(i))
498 bb(i)=1.0d0/dt + vvt*s2/del + 2.0d0*dtracer(i)/drp
499 cc(i)=-vvt*s3/del + 2.0d0*dtracer(i)/(drs*dr(i-1))
500 else
501 aa(i)=-vv*s1/del + 2.0d0*d(i)/(drs*dr(i))
502 bb(i)=1.0d0/dt + vv*s2/del + 2.0d0*d(i)/drp + kprime
503 cc(i)=-vv*s3/del + 2.0d0*d(i)/(drs*dr(i-1))
504 endif
505 dd(i)=c(isp,i)/dt + rxn ! This is where the user-defined reactions go ...
506 !(the coefficient D of the Thomas Algorithm)
507 if (isorp(isp).ge.4) dd(i)=dd(i) + (rhob/por)*alpha(isp)*s(isp,i)
508 if (isorp(isp).ge.4) bb(i)=bb(i) + (rhob/por)*alpha(isp)*((1.0d0-frac(isp))*kd(isp))

509 return
510 end
511 !-----
512 subroutine coef3
513 implicit real *8 (a-h,o-z)
514 PARAMETER (mr=10001,ncomp=10, ns=10)
515 real *8 k1,k2, kd, kprime, kprime2,rxn, uk
516 integer isorp, nr, i, iadvect, userxn
517 common/param1/ r(mr),dr(mr),c(ncomp,mr),cold(ncomp,mr),z1(mr),z2(mr),aa(mr),bb(mr),cc(mr),&
518 dd(mr),s(ncomp,mr),sold(ncomp,mr),v(mr), vt(mr), d(mr),dtracer(mr), qinj, qext,&
519 tinj, trest, text, baquifer, por, rhob, rw,rmax,dt,qinj2,tinj2,tmax,&
520 kd(ncomp), a(ncomp),b(ncomp), alpha(ncomp), frac(ncomp), drone,grat,tau,rdiag,alphan,&
521 c0(ncomp), c1(ncomp),kprime, k(ns),korder(ns),tcrit(ns),time,time_since_inj,&
522 vol_extr_by_vol_inj, Ak, cone, am, q1r, qmr, pi, flag, flag2, q, retard
523 common/param2/nr,idiag, ichasr,isorp(ncomp),imesh,numsp, nseg,i,isp,itr,iadvect,nt,l1,lm,userxn
524 common/userdefparam/uk(10)

525 if (userxn.eq.1) then
526 call reactions(rxn)
527 else
528 rxn = 0.0d0
529 endif

530 vr=0.5d0*(v(i)+v(i+1))
531 vl=0.5d0*(v(i)+v(i-1))
532 vrt=0.5d0*(vt(i)+vt(i+1))
533 vlt=0.5d0*(vt(i)+vt(i-1))
534 if ((vr.gt.0.0d0).and.(vl.gt.0.0d0)) then
535 s1=0.0d0
536 s2=1.0d0
537 s3=0.0d0
538 s4=-1.0d0
539 del = dr(i-1) !0.5d0*(dr(i-1)+dr(i))
540 vv= vr
541 vvt= vrt
542 elseif ((vr.lt.0.0d0).and.(vl.lt.0.0d0)) then

```

```

543 del = dr(i) !0.5d0*(dr(i+1)+dr(i))
544 s1=1.0d0
545 s2=0.0d0
546 s3=-1.0d0
547 s4=0.0d0
548 vv= vl
549 vvt= vlt
550 else
551 del = dr(i) !0.5d0*(dr(i+1)+dr(i))
552 s1=1.0d0
553 s2=0.0d0
554 s3=-1.0d0
555 s4=0.0d0
556 vv= vl
557 vvt= vlt
558 endif

559 drs=dr(i)+dr(i-1)
560 drp=dr(i)*dr(i-1)
561 !-----
562 ! Setup the simultaneous equations
563 !-----
564 if (isp.eq.1) then
565 aa(i)=-vvt*s1/del + 2.0d0*dtracer(i)/(drs*dr(i))
566 bb(i)=1.0d0/dt + vvt*s2/del + vvt*s3/del + 2.0d0*dtracer(i)/drp
567 cc(i)=-vvt*s4/del + 2.0d0*dtracer(i)/(drs*dr(i-1))
568 else
569 aa(i)=-vv*s1/del + 2.0d0*d(i)/(drs*dr(i))
570 bb(i)=1.0d0/dt + vv*s2/del + vv*s3/del + 2.0d0*d(i)/drp + kprime
571 cc(i)=-vv*s4/del + 2.0d0*d(i)/(drs*dr(i-1))
572 endif
573 dd(i)=c(isp,i)/dt + rxn ! This is where the user-defined reactions go

574 if (isorp(isp).ge.4) dd(i)=dd(i) + (rhob/por)*alpha(isp)*s(isp,i)
575 if (isorp(isp).ge.4) bb(i)=bb(i) + (rhob/por)*alpha(isp)*((1.0d0-frac(isp))*kd(isp))

576 return
577 end
578 !-----
579 ! uniform mesh
580 !-----
581 subroutine umesh(rw,n,ymax,dy,y)
582 implicit real *8 (a-h,o-z)
583 PARAMETER (mr=10001,ncomp=10, ns=10)
584 dimension dy(mr),y(mr)
585 del=ymax/float(n-1)
586 y(1)=rw
587 dy(1)=del
588 do 1 i=2,n
589 dy(i)=del
590 y(i)=y(i-1)+del
591 1 continue
592 return
593 end
594 !-----
595 ! non-uniform mesh : geometrically varying

```

```

596 |-----
597 subroutine gmesh(rw,drone,grat,nr,dr,r)
598 implicit real *8 (a-h,o-z)
599 PARAMETER (mr=10001,ncomp=10, ns=10)
600 dimension dr(mr),r(mr)
601 r(1)=rw
602 nrr=nr-1
603 dr(1)=drone
604 rinf=r(1)+drone*(grat**nrr-1.0d0)/(grat-1.0d0)
605 do 1 j=2,nr
606 dr(j)=grat*dr(j-1)
607 1 continue
608 do 3 j=2,nr
609 3 r(j)=r(j-1)+dr(j-1)
610 return
611 end
612 |-----
613 subroutine vmesh(rw,tau,rmax,rc,nr,dr,r)
614 !generates a variable mesh according to equation (16) in the paper
615 implicit real *8 (a-h,o-z)
616 PARAMETER (mr=10001,ncomp=10, ns=10)
617 dimension drr(mr),rr(mr), dr(mr), r(mr)
618 rmax=rmax-rw
619 rc=rc-rw
620 del=1.0d0/dfloat(nr-1)
621 rr(1)=0;
622 anr = 1.0d0 + (dexp(tau)-1.0d0)*(rc/rmax)
623 adr = 1.0d0 + (exp(-tau)-1.0d0)*(rc/rmax)
624 b = (1.0d0/(2.0d0*tau))*dlog(anr/adr)
625 do i=2,nr
626 rr(i)=rr(i-1)+del
627 enddo
628 r(1)=0.0d0
629 do i=2,nr
630 anum = dsinh(tau*(rr(i)-b))
631 aden = dsinh(tau*b)
632 r(i)=1.0 + anum/aden
633 r(i)=rc*r(i)
634 enddo
635 do i=1,nr
636 r(i)=r(i)+rw
637 enddo
638 do i=2,nr
639 dr(i-1)=r(i)-r(i-1)
640 enddo
641 return
642 end
643 |-----
644 subroutine reactions(rxn)
645 ! User-defined reactions can be specified using this module.
646 implicit real *8 (a-h,o-z)
647 PARAMETER (mr=10001,ncomp=10, ns=10)
648 integer isorp, nr, i, iadvect, userxn
649 real *8 k,korder,tcrit, kd, kprime, rxn, uk
650 character *80 filnam
651 character *3 cisp

```

```

652 character *20 fname
653 common/param1/ r(mr),dr(mr),c(ncomp,mr),cold(ncomp,mr),z1(mr),z2(mr),aa(mr),bb(mr),cc(mr),&
654 dd(mr),s(ncomp,mr),sold(ncomp,mr),v(mr),vt(mr),d(mr),dtracer(mr),qinj,qext,&
655 tinj,trest,text,baquifer,por,rhob,rw,rmax,dt,qinj2,tinj2,tmax,&
656 kd(ncomp),a(ncomp),b(ncomp),alpha(ncomp),frac(ncomp),drone,grat,tau,rdiag,alphan,&
657 c0(ncomp),c1(ncomp),kprime,k(ns),korder(ns),tcrit(ns),time,time_since_inj,&
658 vol_extr_by_vol_inj,Ak,cone,am,q1r,qmr,pi,flag,flag2,q,retard
659 common/param2/nr,idiag,ichasr,isorp(ncomp),imesh,numsp,nseg,i,isp,itr,iadvect,nt,l1,lm,userxn
660 common/userdefparam/uk(10)

661 real *8 ks, kk

662 rxn = 0.0d0
663 ks = 0.1d0
664 kk = 0.2d0

665 uk(1)=kk;
666 uk(2)=ks;

667 rr=r(i)
668 t=time

669 !following code generated using MAPLE version 13:
670 if(isp.eq.1) rxn = -dexp(-dsqrt(rr)*t)*rr**(-0.3D1/0.2D1)*(0.4D1*rr**2 &
671 + 0.2D1*Ak*t + alphan*dabs(Ak/rr)*t + alphan*dabs(Ak/rr)*t**2*dsqrt(rr))/0.4D1

672 if(isp.eq.2) rxn = -kk*(c(1,i)/(ks + c(1,i)))*c(2,i) !Monod reaction term

673 if(isp.eq.2) rxn = rxn -dexp(-dsqrt(rr)*t/pi/0.2D1)*rr**(-0.3D1/0.2D1)* &
674 (0.8D1*rr**2*pi*ks+0.8D1*rr**2*pi*dexp(-dsqrt(rr)*t) + &
675 0.4D1*Ak*t*pi*ks + 0.4D1*Ak*t*pi*dexp(-dsqrt(rr)*t) &
676 + 0.2D1*alphan*dabs(Ak/rr)*t*pi*ks + 0.2D1*alphan*dabs(Ak/rr) &
677 *t*pi*dexp(-dsqrt(rr)*t) + alphan*dabs(Ak/rr)*t**2* &
678 dsqrt(rr)*ks + alphan*dabs(Ak/rr)*t**2*dsqrt(rr)*dexp(-dsqrt( &
679 rr)*t) - 0.16D2*kk*dexp(-dsqrt(rr)*t)*pi**2*rr**(0.3D1 /&
680 0.2D1))/pi**2/(ks + dexp(-dsqrt(rr)*t))/0.16D2

681 !Override the default boundary conditions here if desired:
682 ! this is an analytical solution with function values specified on both ends (Dirichlet BCs)
683 ! so we want to override the boundary conditions specified in the code
684 ! if this is not needed, comment the following lines

685 conc1rw = dexp(-dsqrt(rw)*time);
686 conc2rw = dexp(-dsqrt(rw)*time/pi/0.2D1)
687 c0(1)=conc1rw
688 c0(2)=conc2rw
689 conc1nr = dexp(-dsqrt(r(nr))*time);
690 conc2nr = dexp(-dsqrt(r(nr))*time/pi/0.2D1)

691 if(isp.eq.1) then
692 concnr=conc1nr
693 c(isp,nr)=concnr
694 c(isp,1)=c0(isp)
695 cold(isp,1)=c0(isp)

```

```

696 cold(isp,nr)=conc1nr
697 elseif (isp.eq.2) then
698 concnr=conc2nr
699 c(isp,nr)=conc2nr
700 c(isp,1)=c0(isp)
701 cold(isp,1)=c0(isp)
702 cold(isp,nr)=conc2nr
703 else
704 endif

705 l1=1
706 cone=c0(isp)
707 lm=1
708 am=c(isp,nr)

709 return
710 end

```

7 Nomenclature

a	Coefficient in the Freundlich isotherm, equation (5)
B	Parameter for mesh generation, eq. (35-36)
b	Aquifer thickness
C_0	Concentration (eq. 2) near the well casing during injection (boundary condition)
C_{20}	Concentration (eq. 2) near the well casing during chaser injection (boundary condition)
C_T	Concentration of tracer
C_b	Background concentration in the aquifer before the "push" phase
C	Concentration of the reactive component
E, F, G	Coefficients of the tridiagonal matrix, equation (46)
f	Fraction of sites in equilibrium for the kinetic and two-site sorption models
\mathcal{H}	Heaviside step function. $\mathcal{H}(t - a) = 1$ if $t \geq a$ and zero otherwise
K_d	Distribution Coefficient
m	Coefficient in the Freundlich isotherm, equation (5)
n	Order of the reaction, equation (2)
n_r	Number of grid points
N	Total number of reaction rates in equation (2)
p, q	Coefficients in the Langmuir isotherm, equation (6)
Q	Pumping rate
r	Radial coordinate
r'	A uniform grid coordinate used in equation (36) $r' = 1/n_r$
r_c	Radial coordinate where details need to be resolved using fine grids
r_w	Radius of the well casing
r_{max}	Size of the computational domain (distance between well casing and the outer domain)
R	Right hand side of the linear system of equations, equation (46)
R, R_F, R_L	Retardation coefficients for the linear equilibrium, Freundlich and Langmuir isotherms
S	Sorbed-phase concentration
$S_1 \cdots S_4$	Switches (variables that can take only integer values -1, 0, +1) in equation (44)
S_0	Sorbed-phase concentration near the well casing during injection (boundary condition)
S_{20}	Sorbed-phase concentration (boundary condition) during chaser injection

t	Time
t_i^*	Times (since injection) when reaction rates (and possibly orders) change
t_{inj}	Duration of the injection period for the test solution
t_{chaser}	Duration of the injection period for the chaser solution
t_{rest}	Duration of the rest period
t_{ext}	Duration of the extraction period
v	Pore water velocity
v_R, v_L	Velocity at the right and left faces of a control volume around node j . See figure 2
α	First-order kinetic rate parameter, equations (7) and (8)
α_L	Longitudinal dispersivity
γ	Mesh tuning parameter. As γ increases, fine grids are placed around r_c
Δr_j	Step size at the spatial / grid location denoted by j
Δt	Time step
λ	Mesh expansion ratio for the geometric grid (ratio of the geometric progression) eq. (34)
ρ_b	Bulk density

Subscripts & Superscripts

ℓ	Time-level. $\ell + 1$ and ℓ denote the new and old time-levels respectively
L, R	Denote left and right relative to the grid node located at j
j	Index denoting the spatial location

8 References

- Haggerty, R., M.H. Schroth, J.D. Istok, Simplified method of "Push-Pull" test data analysis for determining in situ reaction rate coefficients. *Ground Water*, 36(2), 314-324 (1998)
- Intel® Whitepaper, Quick-Reference Guide to Optimization with Intel® Compilers version 11 For IA-32 processors, Intel® 64 processors and IA-64 processors, Intel Corporation, Santa Clara, CA (2008)
- McGuire, J. T., D.T. Long, M.J. Klug, S.K. Haack, D.W. Hyndman, Evaluating the behavior of oxygen, nitrate, and sulfate during recharge and quantifying reduction rates in a contaminated aquifer, *Environmental Science and Technology*, 36(12), 2693-2700 (2002)
- Pickens, J. F., R.E. Jackson, K.J. Inch, Measurement of distribution coefficients using a radial injection dual-tracer test, *Water Resources Research*, 17(3), 529-544 (1981)
- Phanikumar, M.S. and R.L. Mahajan, Numerical analysis of unsteady thermosolutal convection over a horizontal, isothermal circular cylinder, *Numerical Heat Transfer - Part A: Applications*, Vol. 33, pp. 673-700 (1998)
- Phanikumar, M.S. and J.T. McGuire, A multi-species reactive transport model to estimate biogeochemical rates based on single-well push-pull test data, *Computers & Geosciences*, Vol. 36, No. 8, pp. 997-1004 (2010)
- Roache, P.J. *Computational Fluid Dynamics*, Hermosa Publishers, Albuquerque, NM (1998)
- Roache, P.J. *Fundamentals of Verification and Validation*, Hermosa Publishers, Albuquerque, NM (2009)
- Schroth, M. H., Istok, J. D., Haggerty, R. In situ evaluation of solute retardation using single-well push-pull tests, *Advances in Water Resources*, 24, 105-117 (2001)
- Tannehill, J. C., D.A. Anderson, R.H. Pletcher, *Computational Fluid Mechanics and Heat Transfer*, 337 pp., Taylor & Francis, Washington D.C. (1997).

Van Genuchten, M. Th., R.J. Wagenet, Two-Site/two-region models for pesticide transport and degradation: theoretical development and analytical solutions. *Soil Science Society of America Journal*, 53(5), 1303-1310 (1989).

Waterloo Maple Inc., Maple 13, User Manual, pp. 524 (2009)

If you have questions/comments or if you would like to request new features or report bugs, please e-mail: phani@msu.edu

