PPTEST:
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
Errata

This user manual is written to describe the PPTEST 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:

\[ + \ldots + \left\{ k_{N-1} C_i^{N-1} \begin{array}{ll} t_{N-1}^* \leq t < t_N^* \\ 0 \end{array} \right\} \]

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".

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:

```plaintext
>> 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 $j$, $\Delta r_j$ 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](image-url)  

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 ($c(2,i)$, $c(3,i)$, · · · 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.
\[
\frac{\partial C_T}{\partial t} + v \frac{\partial C_T}{\partial r} = \alpha_L |v| \frac{\partial^2 C_T}{\partial r^2}
\]

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

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:

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

The pore water velocity near the well is given by:

\[
v = \frac{Q}{2\pi b \theta r}
\]

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. PPTES 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 \(S = S_i\).

- **Linear equilibrium sorption:** \(S = K_d C\)
- **Freundlich isotherm:** \(S = a C^{\frac{1}{m}}\)
- **Langmuir isotherm:** \(S = \frac{pqC}{1 + pC}\)
- **One-site kinetic sorption:** \(\frac{\partial S}{\partial t} = \alpha (K_d C - S)\)
- **Two-site kinetic sorption:** \(\frac{\partial S}{\partial t} = \alpha [(1 - f)K_d C - S]\)

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.

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

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.

\[
R = 1 + \frac{\rho_b}{\theta} K_d 
\]

(9)

\[
R_F = 1 + \frac{\rho_b m C^{m-1}}{\theta}
\]

(10)

\[
R_L = 1 + \frac{\rho_b}{\theta} \left( \frac{pq}{(1+pC)^2} \right)
\]

(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.

\[
C(r, t = 0) = C_b : r > r_w
\]

(12)

\[
S(r, t = 0) = S_b : r > r_w
\]

(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:**

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

(14)

\[
S(r_w, t) = S_0 : 0 < t < t_{inj}
\]

(15)

\[
\frac{\partial C(r \to \infty, t)}{\partial r} = 0
\]

(16)

\[
\frac{\partial S(r \to \infty, t)}{\partial r} = 0
\]

(17)

\[
C_{r=r_w} = C_0 : 0 < t < t_{inj}
\]

(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.

\[
\left(-\alpha L \frac{\partial C}{\partial r} + C\right)_{r=r_w} = C_1 : t_{inj} < t < (t_{inj} + t_{chaser}) \\
S(r_w, t) = S_1 : t_{inj} < t < (t_{inj} + t_{chaser})
\]

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

**Rest Period:**

\[
\left(-\alpha L \frac{\partial C}{\partial r} + C\right)_{r=r_w} = 0 : (t_{inj} + t_{chaser}) < t < (t_{inj} + t_{chaser} + t_{rest})
\]

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 \((f = 1 \text{ for the one-site kinetic model})\).

\[
S_0 = (1 - f)K_d C_0 \tag{32}
\]

\[
S_1 = (1 - f)K_d C_1 \tag{33}
\]

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 (\(\Delta r_1\)). \(r_{\text{max}}\) can be calculated using equation (34). The geometric ratio \(\lambda\) that produces a desired \(r_{\text{max}}\) can be easily calculated by taking the logarithms on both sides of equation (34).

\[
r_{\text{max}} = \frac{\Delta r_1 (\lambda^{n_r} - 1)}{(\lambda - 1)} \quad \lambda > 1
\]  

(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 \(r_c\).

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

(35)

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

(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).

\[
\frac{\partial C}{\partial r} \bigg|_j = \frac{C_{j+1}}{\beta_j (\Delta r_j + \Delta r_{j-1})} - \frac{\beta_j C_{j-1}}{\Delta r_j + \Delta r_{j-1}} + \frac{(\Delta r_j - \Delta r_{j-1})}{(\Delta r_j \cdot \Delta r_{j-1})} C_j
\]

(37)

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

(38)

\[
\beta = \left( \frac{\Delta r_j}{\Delta r_{j-1}} \right)
\]

(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).

\[
\frac{\partial}{\partial r} (vC) = \frac{v_R C_R - v_L C_L}{\Delta}
\]

(40)

\[
v_R = \frac{1}{2} \left( v_{j+1} + v_j \right)
\]

(41)

\[
v_L = \frac{1}{2} \left( v_j + v_{j-1} \right)
\]

(42)

\[
\begin{align*}
C_R &= C_j \quad \text{if } v_R > 0 \\
C_R &= C_{j+1} \quad \text{if } v_R < 0 \\
C_L &= C_{j-1} \quad \text{if } v_L > 0 \\
C_L &= C_j \quad \text{if } v_L < 0
\end{align*}
\]

(43)
The above conditions can be combined into one single equation with four switches \( \psi_1, \psi_2, \psi_3 \) and \( \psi_4 \) as shown below.

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

(44)

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

\[
\frac{C_{j+1}^\ell - C_j^\ell}{\Delta t} + \frac{\rho_b}{\theta} \left( \frac{\partial S}{\partial t} \right)_j + \left( \frac{\psi_1 v_R C_{j+1}^\ell + \psi_2 v_R C_j^\ell + \psi_3 v_L C_j^\ell + \psi_4 v_L C_{j-1}^\ell}{\Delta} \right) - \left( \frac{2C_j^\ell}{(\Delta r_j + \Delta r_{j-1})} \right)
\]

\[
= \alpha_L |v|_j^\ell \left\{ \frac{2}{(\Delta r_j + \Delta r_{j-1})} \left[ \frac{C_{j+1}^\ell}{\Delta r_j} - \frac{C_{j-1}^\ell}{\Delta r_{j-1}} \right] - \frac{2C_j^\ell}{(\Delta r_j \cdot \Delta r_{j-1})} \right\} - \sum_{i=1}^{N-1} [\mathcal{H}(t - t_i^*) - \mathcal{H}(t - t_{i+1}^*)] k_i \left[ C_j^{\ell+1} \right]_{n_i} \pm F_j^\ell
\]

(45)

In the above equation, the superscripts \( \ell + 1 \) and \( \ell \) 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:

\[-E_j C_{j-1}^{\ell+1} + F_j C_j^{\ell+1} - G_j C_{j+1}^{\ell+1} = R_j\]

(46)

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

\[
E_j = -\frac{v_R \psi_1}{R \Delta} + \frac{2\alpha_L |v|}{(\Delta r_j + \Delta r_{j-1}) R \Delta r_j}
\]

(47)

\[
F_j = \frac{1}{\Delta t} + \frac{v_R \psi_2 + v_L \psi_3}{R \Delta} + \frac{2\alpha_L |v|}{(\Delta r_j \Delta r_{j-1}) R} + \frac{k_i}{R} \left[ C_j^{\ell} \right]_{n_i-1} + \frac{\rho_b}{\theta} \alpha(1 - f) K_d
\]

(48)

\[
G_j = -\frac{v_L \psi_4}{\Delta R} + \frac{2\alpha_L |v|}{(\Delta r_j + \Delta r_{j-1}) \Delta r_{j-1} R}
\]

(49)

\[
R_j = \frac{C_j^{\ell}}{\Delta t} + \frac{\rho_b}{\theta} \alpha S_j^{\ell} \pm F_j^{\ell}
\]

(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.

\[
\begin{bmatrix}
F_1 & G_1 & 0 & 0 & 0 \\
E_2 & F_2 & G_2 & 0 & 0 \\
E_3 & F_3 & G_3 & 0 & 0 \\
E_4 & F_4 & G_4 & 0 & 0 \\
E_5 & F_5 & G_5 & 0 & 0
\end{bmatrix}
\begin{bmatrix}
C_1^{\ell+1} \\
C_2^{\ell+1} \\
C_3^{\ell+1} \\
C_4^{\ell+1} \\
C_5^{\ell+1}
\end{bmatrix}
= \begin{bmatrix}
R_1 \\
R_2 \\
R_3 \\
R_4 \\
R_5
\end{bmatrix}
\]

(51)

In equations (47-50), \( \Delta \) is an appropriate step size (\( \Delta r_j \) or \( \Delta r_{j-1} \)) depending on the direction of velocity and \( R \) 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 \( R = 1 \). If the nonlinear equilibrium models are used (i.e., \( R \neq 1 \)), then \( \delta = 0 \) in equations (48) and (50). In equation (48), the reaction term was linearized and approximated as:
\[
\frac{k_i}{\mathcal{R}} (C_j^{t+1})_{n_i} = \frac{k_i}{\mathcal{R}} [(C_j^t)^{n_i-1} \cdot (C_j^{t+1})]
\] (52)

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

\[
\frac{S_j^{t+1} - S_j^t}{\Delta t} = \alpha (1 - f) K_d C_j^t - S_j^t
\] (53)

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

\[
E_j = 0
\] (54)

\[
F_j = \frac{1}{\Delta t} + 1
\] (55)

\[
G_j = 0
\] (56)

\[
R_j = \frac{S_j^t}{\Delta t} + \alpha (1 - f) K_d C_j^t
\] (57)

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 \(\beta\) that produces the desired resolution at a given location \(r_c\), the following MATLAB script can be used.

```matlab
clear; clc;
rw=0.104;
rm=50.0;
nr=3001;
tau=10.0; % this is gamma in eqs. (35-36)
rc=0.36-rw;
rm=rm-rw;
%Calculate constants
anr = 1.0 + (exp(tau)-1.0)*(rc/rm);
adr = 1.0 + (exp(-tau)-1.0)*(rc/rm);
b = (1.0/(2.0*tau))*log(anr/adr);
%First generate a uniform mesh
del=1.0/(nr-1);
del=1.0/(nr-1);
rr(1)=0;
for i=2:nr
    rr(i)=b+(1.0/tau)*asinh((rat-1.0d0)*dsinh(tau*b));
    rr(i)=rr(i-1)+del;
```

Figure 2: Nomenclature for the finite-difference stencil
Now generate the variable mesh:

```matlab
r(1)=0;
for i=2:nr
    anum = sinh(tau*(rr(i)-b));
    aden = sinh(tau*b);
    r(i)=1.0 + anum/aden;
    r(i)=rc*r(i);
end
for i=1:nr
    r(i)=r(i)+rw;
end
for i=2:nr
    dr(i-1)=r(i)-r(i-1);
end
dr(nr)=dr(nr-1);
[r(1) dr(1)]
figure(1);
i=1:nr;
loglog(r,dr,'-');
xlabel('Radial distance from well casing, r_{j}');
ylabel('Step size, \Delta r_{j}');
grid on;
```

Figure 3: Examples of variable meshes generated by equations (35-36) for different values of the tuning parameter, $\gamma$. The resolution around the radial location $r_c$ increases with $\gamma$. 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: \( r_{\text{max}}, \, n_r, \, \Delta t \)

Line 2: \( i_{\text{mesh}} \)

Line 3: \( \Delta r_1, \, \lambda \)

Line 4: \( i_{\text{advect}} \)

Line 5: \( \gamma, \, r_c \)

Line 6: \( \alpha, \, \kappa, \, k_d \)

Line 7: \( q_{\text{inj}}, \, q_{\text{ext}}, \, q_{\text{treat}}, \, q_{\text{text}}, \, q_{\text{baquifer}}, \, p_{\text{or}}, \, \alpha_{\text{eral}}, \, \rho_{\text{b}}, \, \rho_w \)

Line 8: \( q_{\text{inj2}}, \, q_{\text{treat2}} \)

Line 9: \( n_{\text{numsp}}, \, n_{\text{seg}} \)

Line 10: \( c_0 \)

Line 11: \( k_{\text{order}} \)

Line 12: \( t_{\text{crit}} \)

Line 13: \( k \)

Line 14: \( c_{1} \)

Line 15: \( \text{isorp} \)

Line 16: \( k_d \)

Line 17: \( \text{usrxn} \)

A detailed explanation of these input parameters is given below.

Line 1: \( r_{\text{max}}, \, n_r, \, \Delta t \)

1. \( r_{\text{max}} \): Distance from the well casing to the outer boundary \( (r_{\text{max}}) \)

2. \( n_r \): Number grid points \( n \)

3. \( \Delta t \): Time step \( \Delta t \)

Line 2: \( i_{\text{mesh}} \)

This parameter is used to select a method for generating the computational mesh. If \( i_{\text{mesh}} = 1 \), 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 \( i_{\text{mesh}} = 2 \), then a variable mesh in which the step size increases geometrically from the well casing will be generated. Parameters \( \Delta r_1, \, \lambda \) will be read from line 3.

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

Line 4: \( i_{\text{advect}} \)

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: idia**
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 ichaser = 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, numsp)

**Line 11: korder**
Order of the reaction for the j-th segment, n_j. A total of nseg values are expected. PPTEST 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: \( t_{crit} \)
The times \( t^*_j \) at which slopes change as shown in Figure 4. A total of \( n_{seg} \) 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 \( n_{seg} \) 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: \( c_1 \)
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 \( k_d \) 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 \( alpha, k_d \) (equation 7) are read from line 16.
If \( isorp = 5 \), then the two-site kinetic sorption model is used. For this case, three parameters \( alpha, frac, k_d \) (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 Example 1: Comparison with the analytical solution of Gelhar & Collins (1971)

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

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

(58)

\[
r_{\text{max}} = \sqrt{\frac{Q t_{\text{inj}}}{\pi b \theta R}}
\]

(59)

where \( V \) denotes the cumulative extracted volume (\( = |Q_{\text{ext}}| t \)), \( V_{\text{inj}} = Q_{\text{inj}} t_{\text{inj}} \) and \( t_{\text{inj}} \) 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 \( \epsilon = \alpha_L/2r_{\text{max}} \). A numerical solution was computed using PPTEST for the following set of parameters. The analytical solution is computed using the MATLAB script.
MATLAB script to compare the numerical solution with the analytical solution of Gelhar & Collins (1971):

```matlab
%----------parameters-------------------
qinj = 2.587; qext=2.282;b=8.0; por=0.38;
rw= 0.052; tinj=94.32; text=405.6; trest=0.00; alphal=0.064;
%----------------------------------------
vij=qinj*tinj;
rmax = sqrt(vinj/(pi*b*por));
epsilon = alphal/(2.0*rmax)
%-----------Compute Approximate Analytical Solution------------------
tt = time; %this is extr_vol_by_inj_vol in PPTEST
term1 = (tt-1.0);
term2 = (abs(1.0-tt)).^0.5;
term2 = term2.*(1.0-tt);
term2 = (2.0-term2);
term2 = ((16.0/3.0)*(alphal/rmax)).*term2;
term2 = term2.^0.5;
!del a1.dat
C = 0.5*erfc(term1./term2);
out = [tt Cnum C];
save -ascii a1.dat out;
plot(tt,C,'b--');
hold on;
plot (tt,Cnum,'r-');
legend('Analytical (Gelhar & Collins (1971))','Numerical');legend boxoff;
xlabel('V/V_{inj}','fontweight','bold','fontsize',12); ylabel('C/C0','rotation',0);
set(gca,'xlim', [0 3]);
%----------------------------------------------------------
```

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 $^{85}\text{Sr}$ using $^{131}\text{I}$ 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 $r_{\text{max}} = 10$ were used. Grid clustering was accomplished using a tuning parameter $\beta = 15$. $^{85}\text{Sr}$ was described using the linear equilibrium sorption model ($\text{isorp} = 1$).

shown below and uses the same values of $\alpha_L$ 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
```
<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.68</td>
<td>0.00636</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>0.78</td>
<td>0.00729</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>0.88</td>
<td>0.00823</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>0.98</td>
<td>0.00917</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>1.08</td>
<td>0.01010</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>1.18</td>
<td>0.01104</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>1.28</td>
<td>0.01197</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>1.38</td>
<td>0.01291</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>1.48</td>
<td>0.01384</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>1.58</td>
<td>0.01478</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>1.68</td>
<td>0.01571</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>1.78</td>
<td>0.01665</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>1.88</td>
<td>0.01758</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>1.98</td>
<td>0.01852</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>2.08</td>
<td>0.01945</td>
<td>0.1000E+0001</td>
</tr>
<tr>
<td>2.18</td>
<td>0.02039</td>
<td>0.1000E+0001</td>
</tr>
</tbody>
</table>

--- --- ---

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>79.08</td>
<td>0.73957</td>
<td>0.83199E+0000</td>
</tr>
<tr>
<td>79.18</td>
<td>0.74051</td>
<td>0.83091E+0000</td>
</tr>
<tr>
<td>79.28</td>
<td>0.74145</td>
<td>0.82983E+0000</td>
</tr>
<tr>
<td>79.38</td>
<td>0.74238</td>
<td>0.82875E+0000</td>
</tr>
<tr>
<td>79.48</td>
<td>0.74332</td>
<td>0.82766E+0000</td>
</tr>
<tr>
<td>79.58</td>
<td>0.74425</td>
<td>0.82657E+0000</td>
</tr>
<tr>
<td>79.68</td>
<td>0.74519</td>
<td>0.82548E+0000</td>
</tr>
<tr>
<td>79.78</td>
<td>0.74612</td>
<td>0.82438E+0000</td>
</tr>
<tr>
<td>79.88</td>
<td>0.74706</td>
<td>0.82328E+0000</td>
</tr>
<tr>
<td>79.98</td>
<td>0.74799</td>
<td>0.82217E+0000</td>
</tr>
<tr>
<td>80.08</td>
<td>0.74893</td>
<td>0.82106E+0000</td>
</tr>
<tr>
<td>80.18</td>
<td>0.74986</td>
<td>0.81995E+0000</td>
</tr>
<tr>
<td>80.28</td>
<td>0.75080</td>
<td>0.81884E+0000</td>
</tr>
<tr>
<td>80.38</td>
<td>0.75173</td>
<td>0.81772E+0000</td>
</tr>
<tr>
<td>80.48</td>
<td>0.75267</td>
<td>0.81660E+0000</td>
</tr>
<tr>
<td>80.58</td>
<td>0.75360</td>
<td>0.81548E+0000</td>
</tr>
<tr>
<td>80.68</td>
<td>0.75454</td>
<td>0.81435E+0000</td>
</tr>
<tr>
<td>80.78</td>
<td>0.75547</td>
<td>0.81322E+0000</td>
</tr>
<tr>
<td>80.88</td>
<td>0.75641</td>
<td>0.81208E+0000</td>
</tr>
<tr>
<td>80.98</td>
<td>0.75734</td>
<td>0.81095E+0000</td>
</tr>
<tr>
<td>81.08</td>
<td>0.75828</td>
<td>0.80981E+0000</td>
</tr>
<tr>
<td>81.18</td>
<td>0.75921</td>
<td>0.80866E+0000</td>
</tr>
<tr>
<td>81.28</td>
<td>0.76015</td>
<td>0.80752E+0000</td>
</tr>
<tr>
<td>81.38</td>
<td>0.76108</td>
<td>0.80637E+0000</td>
</tr>
<tr>
<td>81.48</td>
<td>0.76202</td>
<td>0.80521E+0000</td>
</tr>
<tr>
<td>81.58</td>
<td>0.76296</td>
<td>0.80406E+0000</td>
</tr>
<tr>
<td>81.68</td>
<td>0.76389</td>
<td>0.80290E+0000</td>
</tr>
<tr>
<td>81.78</td>
<td>0.76483</td>
<td>0.80174E+0000</td>
</tr>
<tr>
<td>81.88</td>
<td>0.76576</td>
<td>0.80057E+0000</td>
</tr>
</tbody>
</table>

--- --- ---
<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>403.78</td>
<td>3.7762</td>
<td>0.95051</td>
<td>E-0008</td>
</tr>
<tr>
<td>403.88</td>
<td>3.7771</td>
<td>0.94429</td>
<td>E-0008</td>
</tr>
<tr>
<td>403.98</td>
<td>3.7782</td>
<td>0.93811</td>
<td>E-0008</td>
</tr>
<tr>
<td>404.08</td>
<td>3.7790</td>
<td>0.93198</td>
<td>E-0008</td>
</tr>
<tr>
<td>404.18</td>
<td>3.7799</td>
<td>0.92588</td>
<td>E-0008</td>
</tr>
<tr>
<td>404.28</td>
<td>3.7809</td>
<td>0.91982</td>
<td>E-0008</td>
</tr>
<tr>
<td>404.38</td>
<td>3.7818</td>
<td>0.91380</td>
<td>E-0008</td>
</tr>
<tr>
<td>404.48</td>
<td>3.7827</td>
<td>0.90782</td>
<td>E-0008</td>
</tr>
<tr>
<td>404.58</td>
<td>3.7837</td>
<td>0.90188</td>
<td>E-0008</td>
</tr>
<tr>
<td>404.68</td>
<td>3.7846</td>
<td>0.89598</td>
<td>E-0008</td>
</tr>
<tr>
<td>404.78</td>
<td>3.7856</td>
<td>0.89012</td>
<td>E-0008</td>
</tr>
<tr>
<td>404.88</td>
<td>3.7865</td>
<td>0.88429</td>
<td>E-0008</td>
</tr>
<tr>
<td>404.98</td>
<td>3.7874</td>
<td>0.87851</td>
<td>E-0008</td>
</tr>
<tr>
<td>405.08</td>
<td>3.7884</td>
<td>0.87276</td>
<td>E-0008</td>
</tr>
<tr>
<td>405.18</td>
<td>3.7893</td>
<td>0.86705</td>
<td>E-0008</td>
</tr>
<tr>
<td>405.28</td>
<td>3.7902</td>
<td>0.86138</td>
<td>E-0008</td>
</tr>
<tr>
<td>405.38</td>
<td>3.7912</td>
<td>0.85574</td>
<td>E-0008</td>
</tr>
<tr>
<td>405.48</td>
<td>3.7921</td>
<td>0.85015</td>
<td>E-0008</td>
</tr>
<tr>
<td>405.58</td>
<td>3.7930</td>
<td>0.84459</td>
<td>E-0008</td>
</tr>
<tr>
<td>405.68</td>
<td>3.7940</td>
<td>0.83906</td>
<td>E-0008</td>
</tr>
</tbody>
</table>

out2.dat:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.08</td>
<td>0.0075</td>
<td>0.10000</td>
<td>E+0001</td>
</tr>
<tr>
<td>0.18</td>
<td>0.0168</td>
<td>0.10000</td>
<td>E+0001</td>
</tr>
<tr>
<td>0.28</td>
<td>0.0262</td>
<td>0.10000</td>
<td>E+0001</td>
</tr>
<tr>
<td>0.38</td>
<td>0.0355</td>
<td>0.10000</td>
<td>E+0001</td>
</tr>
<tr>
<td>0.48</td>
<td>0.0449</td>
<td>0.10000</td>
<td>E+0001</td>
</tr>
<tr>
<td>0.58</td>
<td>0.0542</td>
<td>0.10000</td>
<td>E+0001</td>
</tr>
<tr>
<td>0.68</td>
<td>0.0636</td>
<td>0.10000</td>
<td>E+0001</td>
</tr>
<tr>
<td>0.78</td>
<td>0.0729</td>
<td>0.10000</td>
<td>E+0001</td>
</tr>
<tr>
<td>0.88</td>
<td>0.0823</td>
<td>0.10000</td>
<td>E+0001</td>
</tr>
<tr>
<td>0.98</td>
<td>0.0917</td>
<td>0.10000</td>
<td>E+0001</td>
</tr>
<tr>
<td>1.08</td>
<td>0.1010</td>
<td>0.10000</td>
<td>E+0001</td>
</tr>
<tr>
<td>1.18</td>
<td>0.1104</td>
<td>0.99999</td>
<td>E+0000</td>
</tr>
</tbody>
</table>

70.58 | 0.66008 | 0.74186 | E+0000                  |
| 70.68 | 0.6612 | 0.74101 | E+0000                  |
| 70.78 | 0.66195| 0.74016 | E+0000                  |
| 70.88 | 0.66298| 0.73931 | E+0000                  |
| 70.98 | 0.66382| 0.73846 | E+0000                  |
| 71.08 | 0.66476| 0.73761 | E+0000                  |
| 71.18 | 0.66569| 0.73676 | E+0000                  |
| 71.28 | 0.66663| 0.73591 | E+0000                  |
| 71.38 | 0.66756| 0.73506 | E+0000                  |
| 71.48 | 0.66850| 0.73420 | E+0000                  |
| 71.58 | 0.66943| 0.73335 | E+0000                  |
| 71.68 | 0.67037| 0.73249 | E+0000                  |
| 71.78 | 0.67130| 0.73164 | E+0000                  |
| 71.88 | 0.67224| 0.73078 | E+0000                  |
5.3 Example 3: 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.
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:**

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Ratio</th>
<th>Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00300</td>
<td>0.00152</td>
<td>0.81999E+0002</td>
</tr>
<tr>
<td>0.01300</td>
<td>0.00659</td>
<td>0.81999E+0002</td>
</tr>
<tr>
<td>0.02300</td>
<td>0.01167</td>
<td>0.81999E+0002</td>
</tr>
<tr>
<td>0.03300</td>
<td>0.01674</td>
<td>0.81999E+0002</td>
</tr>
<tr>
<td>0.04300</td>
<td>0.02181</td>
<td>0.83525E+0002</td>
</tr>
<tr>
<td>0.05300</td>
<td>0.02688</td>
<td>0.84917E+0002</td>
</tr>
<tr>
<td>0.06300</td>
<td>0.03195</td>
<td>0.86188E+0002</td>
</tr>
<tr>
<td>0.07300</td>
<td>0.03702</td>
<td>0.87348E+0002</td>
</tr>
<tr>
<td>0.08300</td>
<td>0.04210</td>
<td>0.88408E+0002</td>
</tr>
<tr>
<td>0.09300</td>
<td>0.04717</td>
<td>0.89375E+0002</td>
</tr>
<tr>
<td>0.10300</td>
<td>0.05224</td>
<td>0.90260E+0002</td>
</tr>
<tr>
<td>0.11300</td>
<td>0.05731</td>
<td>0.91068E+0002</td>
</tr>
<tr>
<td>0.12300</td>
<td>0.06238</td>
<td>0.91806E+0002</td>
</tr>
<tr>
<td>0.13300</td>
<td>0.06746</td>
<td>0.92481E+0002</td>
</tr>
<tr>
<td>0.14300</td>
<td>0.07253</td>
<td>0.93097E+0002</td>
</tr>
<tr>
<td>0.15300</td>
<td>0.07760</td>
<td>0.93661E+0002</td>
</tr>
<tr>
<td>0.16300</td>
<td>0.08267</td>
<td>0.94176E+0002</td>
</tr>
<tr>
<td>0.17300</td>
<td>0.08774</td>
<td>0.94647E+0002</td>
</tr>
<tr>
<td>0.18300</td>
<td>0.09281</td>
<td>0.95077E+0002</td>
</tr>
<tr>
<td>0.19300</td>
<td>0.09789</td>
<td>0.95469E+0002</td>
</tr>
<tr>
<td>0.20300</td>
<td>0.10296</td>
<td>0.95827E+0002</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Ratio</th>
<th>Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.57300</td>
<td>1.81216</td>
<td>0.16269E+0001</td>
</tr>
<tr>
<td>3.58300</td>
<td>1.81723</td>
<td>0.15941E+0001</td>
</tr>
<tr>
<td>3.59300</td>
<td>1.82230</td>
<td>0.15619E+0001</td>
</tr>
<tr>
<td>3.60300</td>
<td>1.82737</td>
<td>0.15304E+0001</td>
</tr>
<tr>
<td>3.61300</td>
<td>1.83245</td>
<td>0.14994E+0001</td>
</tr>
<tr>
<td>3.62300</td>
<td>1.83752</td>
<td>0.14691E+0001</td>
</tr>
<tr>
<td>3.63300</td>
<td>1.84259</td>
<td>0.14393E+0001</td>
</tr>
<tr>
<td>3.64300</td>
<td>1.84766</td>
<td>0.14101E+0001</td>
</tr>
</tbody>
</table>

**out2.dat:**

<table>
<thead>
<tr>
<th>Time (s)</th>
<th>Ratio</th>
<th>Concentration</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00300</td>
<td>0.00152</td>
<td>0.16400E+0002</td>
</tr>
<tr>
<td>0.01300</td>
<td>0.00659</td>
<td>0.16400E+0002</td>
</tr>
<tr>
<td>0.02300</td>
<td>0.01167</td>
<td>0.16400E+0002</td>
</tr>
<tr>
<td>0.03300</td>
<td>0.01674</td>
<td>0.16400E+0002</td>
</tr>
<tr>
<td>0.04300</td>
<td>0.02181</td>
<td>0.16705E+0002</td>
</tr>
<tr>
<td>0.05300</td>
<td>0.02688</td>
<td>0.16983E+0002</td>
</tr>
<tr>
<td>0.06300</td>
<td>0.03195</td>
<td>0.17238E+0002</td>
</tr>
<tr>
<td>0.07300</td>
<td>0.03702</td>
<td>0.17470E+0002</td>
</tr>
<tr>
<td>0.08300</td>
<td>0.04210</td>
<td>0.17682E+0002</td>
</tr>
<tr>
<td>0.09300</td>
<td>0.04717</td>
<td>0.17875E+0002</td>
</tr>
</tbody>
</table>
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 \( r_{\text{max}} \) of 10 and a porosity of 0.33. The diagnostic node was placed at a radial distance of 0.15 (\( \text{id} = 8 \) for \( n = 501 \) and \( \text{id} = 43 \) 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`.

\[
\frac{\partial C_1}{\partial t} + \frac{A}{r} \frac{\partial C_1}{\partial r} = \alpha_L \frac{A}{r} \frac{\partial^2 C_1}{\partial r^2} + S_1 \tag{60}
\]

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

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

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

The above system of equations has the following analytical solutions:

\[
C_1 = e^{-t\sqrt{r}} \tag{64}
\]

\[
C_2 = e^{-\frac{t\sqrt{r}}{2\pi}} \tag{65}
\]

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 PPTEST has the ability to simulate coupled system of equations such as (60-61) based on Monod or Michels-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 PPTEST 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

```fortran
program ppt
  implicit real *8 (a-h,o-z)
  !There are 5 lines similar to the one below in the code. Change the numbers here
  !if you want to increase the grid size (mr), the number of components simulated (ns)
  !or the number of piecewise-linear segments used to describe the data
  !on a plot of log(c_r/c_tr) versus time since injection. Here c_r is the reactive component
  !and c_tr is the tracer.
  PARAMETER (mr=10001,ncomp=10, ns=10)
  !-------------------------------------------------------------------------
  !Written by Mantha Phanikumar, Michigan State University (phani@msu.edu)
  !DISCLAIMER: This software is distributed in the hope that it will be useful.
  !Author makes no warranty that the software will meet your requirements
  !and shall not be held liable for any damages resulting from the correct
  !or incorrect use of the software. The software and its documentation
  !could include technical or other mistakes, inaccuracies or typographical
  !errors. Efforts will be made to correct any errors in future versions.
  !-------------------------------------------------------------------------
  ! use any consistent system of units. no unit conversions in code.
  real *8  kd, k, korder, kprime, canaly(10)
  integer isorp, nr, i, iadvect, userxn
  character *80 filnam
  common/param1/r(mr),dr(mr),c(ncomp,mr),cold(ncomp,mr),z1(mr),z2(mr),aa(mr),bb(mr),cc(mr),
  dd(mr),s(ncomp,mr),sold(ncomp,mr),v(mr), vt(mr), d(mr),dtracer(mr),qinj,qext,&
  tinj, trest, text, baquifer, por, rhob, rw,rmax,dt,qinj2,tinj2,tmax,&
  kd(ncomp), a(ncomp),b(ncomp), alpha(ncomp), frac(ncomp), drone,grat,tau,rdiag,alphal,&
  c0(ncomp), c1(ncomp),kprime, k(ns),korder(ns),tcrit(ns),time,time_since_inj,&
```

write ('*,*) 'what is the name of the file (input parameters will be read from this file)?'
read (*.*) filnam
open(unit=1,file=filnam)

!-------------------------------------
! Read mesh-related parameters
!-------------------------------------
drone=0.0d0; grat=0.0d0; tau=0.0d0; rdiag=0.0d0;
do i=1,ncomp
isorp(i)=0
enddo

read (1,*) rmax, nr, dt
read (1,*) imesh
if (imesh.eq.2) read (1,*) drone,grat
if (imesh.eq.3) read (1,*) tau, rdiag
read (1,*) iadvect
read (1,*) idiag
read (1,*) ichasr

!-----------------------------------------
!Injection, extraction and other parameters
!------------------------------------------
read (1,*) qinj, qext, tinj, trest, text, baquifer,por,alphal,rhob,rw
if(ichasr.eq.1) then
read (1,*) qinj2, tinj2
else
qinj2=0.0d0
tinj2=0.0d0
endif

!-----------------------------------------
!Number of species, number of piece-wise linear segments
!------------------------------------------
read (1,*) numsp, nseg
if (numsp.eq.0) then
write(*,*) 'Number of species is zero! STOP.'
STOP
endif

read (1,*) (c0(i), i=1,numsp)
if (nseg.gt.0) then
read (1,*) (korder(i),i=1,nseg)
read (1,*) (tcrit(i), i=1,nseg)
read (1,*) (k(i), i=1,nseg)
endif

!-----------------------------------------
!input chaser concentration
if(ichasr.eq.1) then
read (1,*) (c1(i), i=1,numsp)
endif

a=0.0d0; b=0.0d0; alpha=0.0d0; frac=0.0d0;
read (1,*) (isorp(i),i=1,numsp)
do i = 1,numsp
  if (isorp(i).eq.1) read (1,*) kd(i)        !linear sorption
  if (isorp(i).eq.2) read (1,*) a(i),b(i)    !freundlich isotherm
  if (isorp(i).eq.3) read (1,*) a(i),b(i)    !langmuir isotherm
  if (isorp(i).eq.4) read (1,*) alpha(i),kd(i) !one-site kinetic sorption
  if (isorp(i).eq.5) read (1,*) alpha(i),frac(i),kd(i) !two-site kinetic sorption
endo

read (1,*) userxn

close(unit=1)

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

!---------------------------------------------------------------------
!the times tcrit(1) and tcrit(2) where slopes change (on a plot of
![log(c_r/c_tr) vs t]) are counted from the beginning of injection.
pi = 4.0d0*datan(1.0d0)

!Generate a log file with all input data
open(unit=1, file = 'log.txt')
write (1,*) 'rmax, nr, dt:', rmax, nr, dt
write (1,*) 'imesh:', imesh
if (imesh.eq.2) write (1,*) 'drone,grat:', drone,grat
if (imesh.eq.3) write (1,*) 'tau, rdiag:', tau, rdiag
write (1,*) 'iadvect:', iadvect
write (1,*) 'idiag:', idiag
write (1,*) 'ichasr:', ichasr
write (1,*) 'qinj, qext, tinj, trest, text, baquifer,por,alphal,rhob,rw:'
write (1,*) qinj, qext, tinj, trest, text, baquifer,por,alphal,rhob,rw
if(ichasr.eq.1) then
  write (1,*) 'qinj2, tinj2:'
  write (1,*) qinj2, tinj2
endif
PPTEST: User Manual

write (1,*) 'numsp, nseg:
write (1,*) numsp, nseg
!------------------------------------------
write(1,*) 'c0 values for all species:
write(1,*) (c0(i), i=1,numsp)
if (nseg.gt.0) then
write (1,*) 'korder for all piece-wise linear segments:
write (1,*) (korder(i),i=1,nseg)
write (1,*) 'times (tcrit) at which slopes change for all segments (tcrit(1) = 0):
write (1,*) (tcrit(i), i=1,nseg)
write(1,*) 'reaction ks for all segments:
write (1,*) (k(i), i=1,nseg)
endif
!input chaser concentration
if(ichasr.eq.1) then
write (1,*) 'chaser concentration for all species:
write (1,*) (c1(i), i=1,numsp)
endif
write (1,*) 'Sorption-related parameters:
write(1,*) 'isorp for all species:
write (1,*) (isorp(i),i=1,ncomp)
do i=1,numsp
if (isorp(i).eq.1)then
write (1,*) 'isp, isorp(isp), kd(isp): ', i,isorp(i),kd(i)
endif
if (isorp(i).eq.2) then
write (1,*) 'isp, isorp(isp), a(isp), b(isp):', i, isorp(i),a(i),b(i)
endif
if (isorp(i).eq.3) then
write (1,*) 'isp, isorp(isp), a(isp), b(isp):', i, isorp(i),a(i),b(i)
endif
if (isorp(i).eq.4) then
write (1,*) 'isp, isorp(isp), alpha(isp), kd(isp):',i,isorp(i),alpha(i),kd(i)
endif
if (isorp(i).eq.5) then
write (1,*) 'isp, isorp(isp), alpha(isp), frac(isp), kd(isp):',i,isorp(i),alpha(i),frac(i),kd(i)
endif
enddo
!input chaser concentration
if (rw.eq.0.0) then
write(*,*)
write(*,*) 'error: radius of well casing is zero (rw = 0). stop
write(*,*)
stop
endif
if (ichasr.eq.0) then
qinj2=0.0d0
tinj2=0.0d0
endif
tinjtotal = tinj + tinj2
nrr=nr-1
iref = 2
if (imesh.eq.1) then
call umesh(rw,nr,rmax,dr,r)
elseif (imesh.eq.2) then
call gmesh(rw,drone,grat,nr,dr,r)
elseif (imesh.eq.3) then
    call vmesh(rw, tau, rmax, rdiag, nr, dr, r)
endif

open (unit=3, file='mesh.txt')
do i=1,nr
    write(3,*) i, dr(i), r(i)
endo

open (unit=3, file='mesh.txt')
do i=1,nr
    write(3,*) i, dr(i), r(i)
endo

c---------initialize variables-------------------
do isp=1,numsp
    do 16 i=2,nr
        c(isp,i)=0
cold(isp,i)=0
        s(isp,i)=0.0d0
        sold(isp,i)=0.0d0
    16 continue
endo
itr=0
time = 0.0d0
ls = 0
nt= (tmax/dt) + 1
do j = 1,nseg
tcrit(j)=tinjtotal+trest+tcrit(j)
endo

1 continue
kount=kount+1
itr=itr+1
time = time + dt
time_since_inj = (time-tinjtotal)
vol_extr_by_vol_inj = time_since_inj*qext/(qinj*tinj + qinj2*tinj2)
do isp = 1,numsp !Solve for all species
    if ((isorp(isp).eq.0).or.(isorp(isp).ge.4)) retard = 1.0d0
    if (isorp(isp).eq.1) retard = 1.0d0 + rhob*kd(isp)/por
    if (isorp(isp).eq. 4) frac(isp) = 1.0d0
    if (isp.eq.1) retard = 1.0d0
    if (isorp(isp).ge.4) then
        retard = 1.0d0
        s0=(1.0d0-frac(isp))*kd(isp)*c0(isp) ! c0 is conc. for first injection
        s1=(1.0d0-frac(isp))*kd(isp)*c1(isp) ! c1 is conc. for chaser
    endif

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

if(time.le.tinj) then
    c(isp,1)=c0(isp)
cold(isp,1)=c0(isp)
    cone = c0(isp)
c(isp,nr)=c(isp,nrr) ! neumann conditon
cold(isp,nr)=cold(isp,nrr)
if(isorp(isp).ge.4) then
        s(isp,1)=s0
        sold(isp,1)=s0
        s(isp,nr)=s(isp,nrr)
        sold(isp,nr)=sold(isp,nrr)
    endif
q=qinj
flag=1.0d0
flag2=1.0d0
elseif (((time.gt.tinj).and.(time.lt.tinjtotal)).and.(ichasr.eq.1)) then
    c(isp,1)=c1(isp)
    cold(isp,1)=c1(isp)
    cone = c1(isp)
    c(isp,nr)=c(isp,nrr) ! neumann conditon
    cold(isp,nr)=cold(isp,nrr)
    if(isorp(isp).ge.4) then
        s(isp,1)=s1
        sold(isp,1)=s1
        s(isp,nr)=s(isp,nrr)
        sold(isp,nr)=sold(isp,nrr)
    endif
    flag=1.0d0
    q=qinj2
    flag2=1.0d0
elseif ((time.gt.tinjtotal).and.(time.lt.(trest+tinjtotal))) then
    ss=alphal/dr(1) ! using mixed boundary conditions (c(1) = cold(2)*ss/(1+ss))
    c(isp,1)=c(isp,2)*(ss/(1.0d0+ss))
    !cold(isp,1)=cold(isp,2)*(ss/(1.0d0+ss))
    c(isp,nr)=c(isp,nrr)
    cold(isp,nr)=cold(isp,nrr)
    if(isorp(isp).ge.4) then
        s(isp,1)=s(isp,2)
        !sold(isp,1)=sold(isp,2)
        s(isp,nr)=s(isp,nrr)
        !sold(isp,nr)=sold(isp,nrr)
    endif
    q=0.0d0 !check
    flag = 0.0d0
    flag2=0.0d0
elseif ((time.gt.(trest+tinjtotal)).and.(time.le.tmax)) then
    c(isp,1)=c(isp,2)
    !cold(isp,1)=cold(isp,2)
    c(isp,nr)=c(isp,nrr) ! neumann conditon
    !cold(isp,nr)=cold(isp,nrr)
    if(isorp(isp).ge.4) then
        s(isp,1)=s(isp,2)
        !sold(isp,1)=sold(isp,2)
        s(isp,nr)=s(isp,nrr)
        !sold(isp,nr)=sold(isp,nrr)
    endif
    flag = -1.0d0
    q=qext
    flag2=1.0d0
endif

!uncomment these lines for example 4 in the paper
!(user-defined reactions with source terms: analytical solution)
!flag = 1.0d0;
!flag2=1.0d0;
!q=qinj;

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

do i=1,nr
  if (isorp(isp).eq.2) retard = 1.0d0 + rhob*a(isp)*b(isp)*c(isp,i)**(b(isp)-1.0d0)/por
  if (isorp(isp).eq.3) retard = 1.0d0 + (rhob/por)*a(isp)*b(isp)/(1.0d0+a(isp)*c(isp,i))**2)
  vt(i)=Ak/r(i)
  v(i)=vt(i)/retard
  dtracer(i)=flag2*alphal*dabs(vt(i))
  d(i)=flag2*alphal*dabs(v(i))
endo

!--------------------------------------------------------------------------
! boundary conditions
!--------------------------------------------------------------------------

if(time.le.tinj) then
  l1=1
  cone=c0(isp)
  lm=2
  am=0.0d0
  q1r=c0(isp)
  qmr=0.0d0 !injection, dirichlet bc
elseif (((time.gt.tinj).and.(time.le.tinjtotal)).and.(ichasr.eq.1)) then
  l1=1;
  cone=c1(isp);
  lm=2;
  am=0.0d0;
  q1r=c1(isp);
  qmr=0.0d0 !injection, chaser, dirichlet
elseif ((time.gt.tinjtotal).and.(time.le.(trest+tinjtotal))) then
  l1=3;
  cone=ss;
  lm=2;
  am=0.0d0;
  q1r=0.0d0;
  qmr=0.0d0 !rest:
elseif ((time.gt.(trest+tinjtotal)).and.(time.le.tmax)) then
  l1=2;
  cone=0.0d0;
  lm=1;
  am=0.0d0;
  q1r=0.0d0;
  qmr=0.0d0
endif

!--------------------------------------------------------------------------
! solve for concentrations:
!--------------------------------------------------------------------------
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 differencing - 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.tinjtotal)).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.tinjtotal).and.(time.le.(trest+tinjtotal))) then
384     l1=2; sone=0.0d0; lm=2; am=0.0d0 ; q1=0.0d0; qm=0.0d0
385  elseif ((time.gt.(trest+tinjtotal)).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
s(isp,jj)=z2(jj)
enddo
endif

!======================================================================================================
!
! writing the output file
!
!======================================================================================================
!
!following 2 lines useful to generate the analytical solution in example 4
!
!c analy(1) = dexp(-dsqrt(r(idiag))*time)
!c analy(2) = dexp(-dsqrt(r(idiag))*time/pi/0.2D1)
!
! The following logic creates as many output files as there are number of species
!
!(one file for each species):
!
write(cisp, '(i3)') isp
fname = 'out'//trim(adjustl(cisp))//'.dat'
open(unit=3,file=fname, access='append')
if (time_since_inj.ge.0.0d0) then
  if (isorp(isp).ge.4) then
    write(3,11) time_since_inj, vol_extr_by_vol_inj, c(isp,idiag), s(isp,idiag)
  else
    write(3,12) time_since_inj, vol_extr_by_vol_inj, c(isp,idiag)
  endif
11 format(f12.5,1x,f12.5, 2x, e15.5e4,2x,e15.5e4)
12 format(f12.5,1x,f12.5, 2x, e15.5e4)
endif
close(unit=3)
do 289 i=1,nr
cold(isp,i)=c(isp,i)
289 continue
if (isorp(isp).ge.4) then
do 290 i=1,nr
  sold(isp,i)=s(isp,i)
290 continue
endif
!
! this is where the isp loop ends
!
if (itr.ge.nt) then
  ls = 1
  itr=0
else
  endif
if(ls.eq. 1) goto 2992
goto 1
2992 stop
end

!-------------------------------------------------
!
subroutine thomas(e,f,g,r,nr,phi,lbc1,valbc1,lbcn,valbcn,q1,qm)
!
implicit real *8 (a-h,o-z)
!
dimension e(nr),f(nr),g(nr),r(nr),a(nr),b(nr),phi(nr)
!
if(lbc1.eq.1) a(1)=0.0d0
if(lbc1.eq.1) b(1)=valbc1
if(lbc1.eq.2) a(1)=1.0d0
if(lbc1.eq.2) b(1)=-valbc1
if(lbc1.eq.3) a(1)=valbc1/(valbc1-1.0d0)
if(lbc1.eq.3) b(1)=q1/(1.0d0-valbc1)
nrr=nr-1
do 1 i=2,nrr
dd=1.0d0/(f(i)-g(i)*a(i-1))
a(i)=e(i)*dd
b(i)=(r(i)+g(i)*b(i-1))*dd
if(lbcn.eq.1) phi(nr)=valbcn
if(lbcn.eq.2) phi(nr)=(b(nrr)+valbcn)/(1.0d0-a(nrr))
if(lbcn.eq.3) phi(nr)=(b(mm)+qm/valbcn)/((1.0d0+valbcn)/valbcn-a(nrr))
do ii=1,nrr
m=nr-ii
phi(m)=a(m)*phi(m+1)+b(m)
enddo
return
end

!-------------------------------------------------
subroutine coef1
!first-order upwind differencing:
implicit real *8 (a-h,o-z)
PARAMETER (mr=10001,ncomp=10, ns=10)
real *8 k1,k2, kd, kprime, kprime2
integer isorp, nr, i, iadvect, userxn

common/param1/ r(mr),dr(mr),c(ncomp,mr),cold(ncomp,mr),z1(mr),z2(mr),aa(mr),bb(mr),cc(mr),
dd(mr),s(ncomp,mr),sold(ncomp,mr),v(mr), vt(mr), d(mr),dtracer(mr), qinj, qext,&
tinj, trest, text, baquifer, por, rhob, rw,rmax,dt,qinj2,tinj2,tmax,&
kpr(ncomp), a(ncomp),b(ncomp), alpha(ncomp), frac(ncomp), drone,grat,tau,rdiag,alphal,&
c0(ncomp), cl(ncomp),kprime, k(ns),korder(ns),tcrit(ns),time,time_since_inj,&
vol_extr_by_vol_inj, Ak, cone, am, qlr, qmr, pi, flag, flag2, q, retard
common/param2/nr,idiag, ichasr,isorp(ncomp),imeshp,numsp, nseg,i,isp,itr,iadvect,nt,11,lm,userxn
common/userdefparam/uk(10)

if (userxn.eq.1) then
call reactions(rxn)
else
rxn = 0.0d0
endif

if (v(i).ge.0.0d0) then
s1=0.0d0
s2=1.0d0
s3=-1.0d0
del = dr(i-1)
vv= v(i)
vvt=vt(i)
else
del = dr(i)
s1=1.0d0
s2=-1.0d0
s3=0.0d0
vv= v(i)
vvt=vt(i)
endif
PPTEST: User Manual

491  \[ \text{drs} = \text{dr}(i) + \text{dr}(i-1) \]
492  \[ \text{drp} = \text{dr}(i) \ast \text{dr}(i-1) \]
493
494  !---------------------------------------------------------------------
495  ! simultaneous equations
496  !---------------------------------------------------------------------
497  if (isp.eq.1) then
498    aa(i) = -\text{vvt} \ast \text{s1}/\text{del} + 2.0d0 \ast \text{tracer}(i)/(\text{drs} \ast \text{dr}(i))
499    bb(i) = 1.0d0/\text{dt} + \text{vvt} \ast \text{s2}/\text{del} + 2.0d0 \ast \text{tracer}(i)/\text{drp}
500    cc(i) = -\text{vvt} \ast \text{s3}/\text{del} + 2.0d0 \ast \text{tracer}(i)/(\text{drs} \ast \text{dr}(i-1))
501  else
502    aa(i) = -\text{vv} \ast \text{s1}/\text{del} + 2.0d0 \ast \text{d}(i)/(\text{drs} \ast \text{dr}(i))
503    bb(i) = 1.0d0/\text{dt} + \text{vv} \ast \text{s2}/\text{del} + 2.0d0 \ast \text{d}(i)/\text{drp} + k\prime
504    cc(i) = -\text{vv} \ast \text{s3}/\text{del} + 2.0d0 \ast \text{d}(i)/(\text{drs} \ast \text{dr}(i-1))
505  endif
506  dd(i) = c(isp,i)/\text{dt} + \text{rxn} ! This is where the user-defined reactions go ...
507  !(the coefficient D of the Thomas Algorithm)
508  if (isorp(isp).ge.4) dd(i) = dd(i) + (\text{rhob}/\text{por}) \ast \alpha(isp) \ast \text{s}(isp,i)
509  if (isorp(isp).ge.4) bb(i) = bb(i) + (\text{rhob}/\text{por}) \ast \alpha(isp) \ast ((1.0d0-\text{frac}(isp)) \ast \text{kd}(isp))
510  return
511  end
512
513  subroutine coef3
514  implicit real *8 (a-h,o-z)
515  parameter (mr=10001,ncomp=10, ns=10)
516  real *8 k1,k2, kd, kprime, kprime2,rxn, uk
517  integer isorp, nr, i, iadvect, userxn
518  common/param1/ r(mr),dr(mr),c(ncomp,mr),cold(ncomp,mr),z1(mr),z2(mr),aa(mr),bb(mr),cc(mr),&
519    dd(mr),s(ncomp,mr),sold(ncomp,mr),v(mr), vt(mr), d(mr),tracer(mr), qinj, qext,&
520    tinj, trest, text, baquifer, por, rhob, rw,rmax,dt,qinj2,tinj2,tmax,&
521    kd(ncomp), a(ncomp),b(ncomp), alpha(ncomp), frac(ncomp), drone,grat,tau,rdiag,alphal,&
522    c0(ncomp), cl(ncomp),kprime, k(ns),korder(ns),tcrit(ns),time,time_since_inj,&
523    vol_extr_by_vol_inj, Ak, cone, am, q1r, qmr, pi, flag, flag2, q, retard
524  common/param2/nr,idiag, ichasr,isorp(ncomp),imesh,numsp, nseg,i,isp,itr,iadvect,nt,11,lm,userxn
525  common/userdefparam/uk(10)
526
527  if (userxn.eq.1) then
528    call reactions(rxn)
529  else
530    rxn = 0.0d0
531  endif
532
533  vr=0.5d0*(v(i)+v(i+1))
534  vl=0.5d0*(v(i)+v(i-1))
535  vrt=0.5d0*(vt(i)+vt(i+1))
536  vlt=0.5d0*(vt(i)+vt(i-1))
537  if ((vr.gt.0.0d0).and.(vl.gt.0.0d0)) then
538    s1=0.0d0
539    s2=1.0d0
540    s3=0.0d0
541    s4=-1.0d0
542    del = dr(i-1)  !0.5d0*(dr(i-1)+dr(i))
543    vv= vr
544    vvt= vrt
545  elseif ((vr.lt.0.0d0).and.(vl.lt.0.0d0)) then
del = dr(i) !0.5d0*(dr(i+1)+dr(i))
s1=1.0d0
s2=0.0d0
s3=-1.0d0
s4=0.0d0
vv= vl
vvt= vlt
else
del = dr(i) !0.5d0*(dr(i+1)+dr(i))
s1=1.0d0
s2=0.0d0
s3=-1.0d0
s4=0.0d0
vv= vl
vvt= vlt
endif
drs=dr(i)+dr(i-1)
drp=dr(i)*dr(i-1)
!--------------------------------------------
! Setup the simultaneous equations
!--------------------------------------------
if (isp.eq.1) then
  aa(i)=-vvt*s1/del + 2.0d0*dtracer(i)/(drs*dr(i))
  bb(i)=1.0d0/dt + vvt*s2/del + vvt*s3/del + 2.0d0*dtracer(i)/drp
  cc(i)=-vvt*s4/del + 2.0d0*dtracer(i)/(drs*dr(i-1))
else
  aa(i)=-vv*s1/del + 2.0d0*d(i)/(drs*dr(i))
  bb(i)=1.0d0/dt + vv*s2/del + vv*s3/del + 2.0d0*d(i)/drp + kprime
  cc(i)=-vv*s4/del + 2.0d0*d(i)/(drs*dr(i-1))
endif
dd(i)=c(isp,i)/dt + rxn ! This is where the user-defined reactions go
if (isorp(isp).ge.4) dd(i)=dd(i) + (rhob/por)*alpha(isp)*s(isp,i)
if (isorp(isp).ge.4) bb(i)=bb(i) + (rhob/por)*alpha(isp)*((1.0d0-frac(isp))*kd(isp))
return
end
!-------------------------------------------------
! uniform mesh
!-------------------------------------------------
subroutine umesh(rw,n,ymax,dy,y)
implicit real *8 (a-h,o-z)
PARAMETER (mr=10001,ncomp=10, ns=10)
dimension dy(mr),y(mr)
del=ymax/float(n-1)
y(1)=rw
dy(1)=del
do 1 i=2,n
dy(i)=del
1 continue
return
end
!-------------------------------------------------
! non-uniform mesh : geometrically varying
subroutine gmesh(rw, drone, grat, nr, dr, r)
implicit real *8 (a-h,o-z)
PARAMETER (mr=10001, ncomp=10, ns=10)
dimension dr(mr), r(mr)
r(1)=rw
nrr=nr-1
dr(1)=drone
rinf=r(1)+drone*(grat**nrr-1.0d0)/(grat-1.0d0)
do 1 j=2, nr
dr(j)=grat*dr(j-1)
1 continue
do 3 j=2, nr
3 r(j)=r(j-1)+dr(j-1)
return
end

subroutine vmesh(rw, tau, rmax, rc, nr, dr, r)
generates a variable mesh according to equation (16) in the paper
implicit real *8 (a-h,o-z)
PARAMETER (mr=10001, ncomp=10, ns=10)
dimension drr(mr), rr(mr), dr(mr), r(mr)
rmax=rmax-rw
rc=rc-rw
del=1.0d0/dfloat(nr-1)
rr(1)=0;
anr = 1.0d0 + (dexp(tau)-1.0d0)*(rc/rmax)
adr = 1.0d0 + (exp(-tau)-1.0d0)*(rc/rmax)
b = (1.0d0/(2.0d0*tau))*dlog(anr/adr)
do i=2, nr
rr(i)=rr(i-1)+del
endo
dr(i)=0.0d0
do i=2, nr
anum = dsinh(tau*(rr(i)-b))
aden = dsinh(tau*b)
r(i)=1.0 + anum/aden
r(i)=rc*r(i)
endo
do i=1, nr
r(i)=r(i)+rw
endo
do i=2, nr
dr(i-1)=r(i)-r(i-1)
endo
return
end

subroutine reactions(rxn)
User-defined reactions can be specified using this module.
implicit real *8 (a-h,o-z)
PARAMETER (mr=10001, ncomp=10, ns=10)
integer isorp, nr, i, iadvect, userxn
real *8 k, korder, tcrit, kd, kprime, rxn, uk
character *80 filnam
character *3 cisp
character *20 fname

common/param1/ r(mr), dr(mr), c(ncomp, mr), cold(ncomp, mr), z1(mr), z2(mr), aa(mr), bb(mr), cc(mr), &
dd(mr), s(ncomp, mr), sold(ncomp, mr), v(mr), vt(mr), d(mr), dtracer(mr), qinj, qext, &
tinj, trest, text, baquifer, por, rhob, rw, rmax, dt, qinj2, tinj2, tmax, &
kd(ncomp), a(ncomp), b(ncomp), alpha(ncomp), frac(ncomp), drone, grat, tau, rdiag, alphal, &
c0(ncomp), c1(ncomp), kprime, k(ns), korder(ns), tcrit(ns), time, time_since_inj, &
vol_extr_by_vol_inj, Ak, cone, am, q1r, qmr, pi, flag, flag2, q, retard

common/param2/nr, idia, ichasr, isorp(ncomp), imesh, numsp, nseg, i, isp, itr, iadvect, nt, 11, lm, userxn
common/userdefparam/uk(10)

real *8 ks, kk

rxn = 0.0d0
ks = 0.1d0
kk = 0.2d0

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

rr = r(i)
t = time

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

if(isp.eq.2) rxn = -kk*(c(1,i)/(ks + c(1,i)))*c2,i ! Monod reaction term

if(isp.eq.2) rxn = rxn -dexp(-dsqrt(rr)*t/0.2D1)**(0.3D1/0.2D1)* &
 (0.8D1*rr**2*pi*ks + 0.8D1*rr**2*pi*dexp(-dsqrt(rr)*t) + &
 0.4D1*Ak*t*pi*ks + 0.4D1*Ak*t*pi*dexp(-dsqrt(rr)*t) &
 + 0.2D1*alphal*dabs(Ak/rr)*t*pi*ks + 0.2D1*alphal*dabs(Ak/rr) &
 *t*pi*dexp(-dsqrt(rr)*t) + alphal*dabs(Ak/rr)*t**2 &
 dsqrt(rr)*ks + alphal*dabs(Ak/rr)*t**2*dsqrt(rr)*dexp(-dsqrt( &
 rr)*t) - 0.16D2*kk*dexp(-dsqrt(rr)*t)*pi**2*rr**(0.3D1 /&
 0.2D1))/pi**2/(ks + dexp(-dsqrt(rr)*t))/0.16D2

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

conc1rw = dexp(-dsqrt(rw)*time);
conc2rw = dexp(-dsqrt(rw)*time/pi/0.2D1)
c0(1) = conc1rw
c0(2) = conc2rw
conc1nr = dexp(-dsqrt(r(nr))*time);
conc2nr = dexp(-dsqrt(r(nr))*time/pi/0.2D1)

if(isp.eq.1) then
concnr = conc1nr
c(isp, nr) = conc1nr
c(isp, 1) = c0(isp)
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, rql. (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
Time
$\times$
Durations (since injection) when reaction rates (and possibly orders) change
Duration of the injection period for the test solution
Duration of the injection period for the chaser solution
Duration of the rest period
Duration of the extraction period
Pore water velocity
Velocity at the right and left faces of a control volume around node $j$. See figure 2
First-order kinetic rate parameter, equations (7) and (8)
Longitudinal dispersivity
Mesh tuning parameter. As $\gamma$ increases, fine grids are placed around $r_c$
Step size at the spatial / grid location denoted by $j$
Time step
Mesh expansion ratio for the geometric grid (ratio of the geometric progression) eq. (34)
Bulk density

Subscripts & Superscripts

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

8 References


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)


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