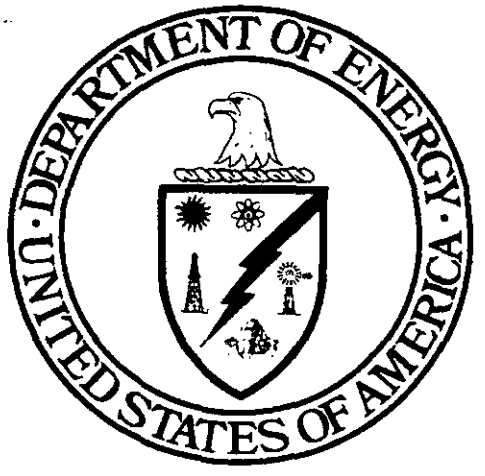
Title 40 CFR Part 191
Compliance Certification
Application
for the
Waste Isolation Pilot Plant

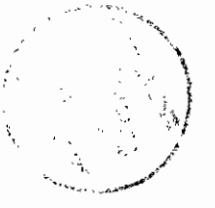
Appendix SECOTP2D



United States Department of Energy
Waste Isolation Pilot Plant

Carlsbad Area Office
Carlsbad, New Mexico

WIPP PA User's Manual for SECOTP2D



SECOTP2D USER'S MANUAL

Version 1.30

Kambiz Salari
Rebecca Blaine

May 8, 1996

Table of Contents

*The User's Manual in Appendix I has its own pagination.

1.0 INTRODUCTION

This document serves as a User's Manual for SECOTP2D, as used in the 1996 WIPP PA calculation. As such, it describes the code's purpose and function, the user's interaction with the code, and the models and methods employed by the code. An example output file is included for the user's convenience.

1.1 Software Identifier

Code Name: SECOTP2D (Sandia-ECOdynamics/TransPort in 2 Dimensions)
WIPP Prefix: ST2D2
Version Number: 1.30, April 18, 1996
Platform: FORTRAN 77 for OpenVMS AXP, ver 6.1, on DEC Alpha

1.2 Points of Contact

1.2.1 Code Sponsor

Rebecca L. Blaine
Ecodynamics Research Associates
P.O. Box 9229
Albuquerque, NM 87119
Voice: (505) 843-7445
Fax: (505) 843-9641

1.2.2 Code Consultant

Kambiz Salari
Ecodynamics Research Associates
P.O. Box 9229
Albuquerque, NM 87119
Voice: (505) 843-7445
Fax: (505) 843-9641

2.0 FUNCTIONAL REQUIREMENTS

R.1 This code performs single component radionuclide transport in fractured or porous media.

R.2 This code performs multiple component radionuclide transport in fractured or porous media.

R.3 This code models single porosity transport in fractured or porous media.

R.4 This code uses a dual porosity model for fractured media with advective and diffusive components in the fractures and only a diffusive component in the matrix material.

R.5 This code models a point source.

R.6 This code calculates the discharge across a user defined boundary.

3.0 REQUIRED USER TRAINING AND/OR BACKGROUND

Code user prerequisites are described in detail in Appendix I.

4.0 DESCRIPTION OF THE MODELS AND METHODS

Models and methods for SECOTP2D are described in detail in Appendix I.

NOTE
Sandia's Configuration Management System (CMS) is the operating system platform for the 1996 WIPP Performance Assessment (PA) calculation. All statements in Appendix I referencing CAM-CON, the operating system upon which CMS is in part based, can also be made about the new CMS operating system.

5.0 CAPABILITIES AND LIMITATIONS OF THE SOFTWARE

Capabilities and limitations of SECOTP2D include the following:

- SECOTP2D can compute multiple component solute transport in fractured porous media using discrete-fracture and dual-porosity models.
- SECOTP2D allows for radioactive decay and generation of daughter products.
- The matrix block equation can model both a matrix material and a clay lining.
- For the fracture-with-matrix block system, transport in the fracture is produced by the combined effect of advection and hydrodynamic dispersion, while transport in the matrix block is assumed to be dominated by molecular diffusion.
- SECOTP2D is an implicit finite volume code that is second-order accurate in space and time. It uses operator splitting, an Approximate Factorization procedure, the delta formulation, and a finite volume staggered mesh.
- SECOTP2D models the advective terms using either Total Variation Diminishing (TVD) or central differencing. TVD eliminates the unwanted oscillations near sharp gradients while maintaining high accuracy in the solution.
- SECOTP2D uses a generalized three-level scheme for time discretization.
- An Approximate Factorization technique is used to alleviate the shortcomings (large memory requirements and CPU time) of banded LU solvers. However, while Approximate Factorization reduces the operation count and memory usage for inverting the coefficient matrix, it does introduce complications in applying the implicit boundary conditions.
- Implicit coupling of the equations for the fracture and matrix block is used. This approach is more robust than explicit coupling.

- The spatial accuracy of the SECOTP2D code, with the TVD option on, is less than second-order accurate, and the deviation from second-order accuracy will depend on how many sharp fronts exist in the computational plane.

- Lack of flow field smoothness near boundaries can pose difficulties for the transport calculation. The SECOTP2D code, in most cases, can eliminate these difficulties by automatically assigning the boundary conditions using the flow field.

- SECOTP2D provides discharge calculations on different predefined, closed boundaries.

These capabilities and limitations of the software are discussed in detail in Appendix I.

6.0 USER INTERACTIONS WITH THE SOFTWARE

User interactions with the software are discussed in detail in Appendix I.

7.0 DESCRIPTION OF INPUT FILES

Input files for SECOTP2D are discussed in detail in Appendix I.

8.0 ERROR MESSAGES

SECOTP2D uses the following lines of code to report error messages. Errors cause program execution to abort.

- These write statements generate an error message indicating that the maximum number of iterations has been exceeded.

```
WRITE(6,*) 'SECO2D-TRANSPORT:'
WRITE(6,*) 'ERROR, Subroutine FRACTURE,  maximum number of '
WRITE(6,*) 'iteration exceeded, MAXITER =',MAXITER
WRITE(6,*) 'It is possible intra-time step iteration is'
WRITE(6,*) 'diverging.'
```

To recover from this error, rerun SECOTP2D with a greater number of time steps.

- These write statements generate an error message indicating that the corner point of discharge surface is not in the computational domain.

```
WRITE(6,*) 'SECO2D-TRANSPORT: ERROR, Subroutine VEL.'
WRITE(6,*) 'Corner point of discharge surface is out of',
   >           ' computational domain.'
```

- These write statements generate an error message indicating that the value of the source function QC cannot be evaluated.

```
WRITE(6,*) 'SECO2D-TRANSPORT: ',
   >        'Error, subroutine QC_INTEGRATE.'
WRITE(6,*) 'value of source function QC cannot be',
```

```
     >         ' evaluated for a specified time.'
  WRITE(6,*)  'TIME = ',TB
```

This error occurs when the time domain of the source function is not aligned with the time domain of the simulation. The error is corrected by ensuring that these time domains are aligned and then rerunning SECOTP2D.

- These write statements generate an error message indicating that location of a source point is not in the computational domain.

```
  WRITE(6,*)  'SECO2D-TRANSPORT: ',
     >              'ERROR, Subroutine SOURCE.'
                 WRITE(6,*) 'Location of a source point is out',
     >              ' of computational domain.'
```

- These write statements generate an error message indicating that an end-of-file has been detected.

```
  WRITE(6,*)  'SECO2D-TRANSPORT:'
              WRITE(6,*) 'ERROR, End-of-file detected (velocity)'
```

9.0 DESCRIPTION OF OUTPUT FILES

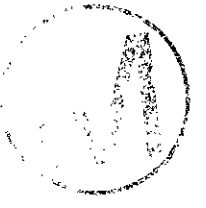
Output files for SECOTP2D are discussed in detail in Appendix I. A sample diagnostics/debug file is provided in Appendix II.

10.0 REFERENCES

References for SECOTP2D are listed in Appendix I.

11.0 APPENDICES

The following section provides the appendices for this document.



ABSTRACT

This is a user's manual for the SECOTP2D (Sandia-ECOdynamics/2-Dimension) code, Version 1.30. This program currently runs on an Alpha AXP computer with OpenVMS 6.1 operating system. SECOTP2D computes multiple component solute transport in fractured porous media using discrete-fracture and dual-porosity models. It allows for radioactive decay and generation of daughter products. In addition, the matrix block equation can model both a matrix material and a clay lining. For the fracture-with-matrix block system, transport in the fracture is produced by the combined effect of advection and hydrodynamic dispersion, while transport in the matrix block is assumed to be dominated by molecular diffusion. SECOTP2D is an implicit finite volume code that is second-order accurate in space and time. It uses operator splitting, an Approximate Factorization procedure, the delta formulation, and a finite volume staggered mesh. The advective terms are modeled using either Total Variation Diminishing (TVD) or central differencing. Time discretization is a generalized three-level scheme. The flow field for transport is obtained from the SECOFL2D code or equivalent. This manual presents the theory, numerical algorithm, and verification of the SECOTP2D code. Also, the pre- and postprocessors are described within the Compliance Assessment Methodology CONtroller (CAMCON) environment.

WIPP PA

User's Manual

for

SECOTP2D, Version 1.30

Document Version 1.01

WPO # 36695

April 18, 1996

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Appendix I: SECOTP2D User's Manual, by Kambiz Salari and Rebecca Blaine

1 INTRODUCTION

The SECOTP2D code, Version 1.30, performs multiple component transport in fractured porous media. The media are represented using discrete-fracture and dual-porosity models. In the discrete-fracture model, the system is assumed to have multiple planar fractures (slab geometry). The governing equations include the effect of advection, hydrodynamic dispersion, linear radioactive decay and generation, and an assumption of linear equilibrium isotherms. In the dual-porosity model, the system is comprised of numerous fractures and a porous rock matrix. The matrix governing equations represent the effect of diffusion, decay, and generation. This equation can model both the porous matrix material and a clay lining. The flow field, which can be either steady or unsteady, is obtained from the SECOFL2D code [1] or equivalent.

SECOTP2D is an implicit finite volume code that is second-order accurate in space and time. It uses operator splitting, an Approximate Factorization procedure, the delta formulation, and a finite volume staggered mesh. The advective terms are modeled using either Total Variation Diminishing (TVD) or central differencing schemes. Time discretization is a generalized three-level scheme. SECOTP2D features manual and automatic boundary definition which can vary from cell to cell. The code has modern FORTRAN structure and due to its operator splitting scheme is quite efficient in execution time and memory usage. The code has an option of computing the history of integrated discharge around any number of defined closed boundaries within the computational mesh.

This manual describes the governing equations, the development of numerical schemes, code verification and machine dependency, and detailed explanations of how to use Version 1.30 of the SECOTP2D the code (pre- and postprocessors) in the environment of the Compliance Assessment Methodology Controller (CAMCON).

The SECOTP2D code is a part of SECO suite of codes that includes flow, transport, and particle tracking in 2- and 3-dimensions. SECOTP2D has been referred to as SECO-TRANSPORT in the open literature [1, 2, 3, 4].

1.1 Code User Prerequisites

In order for the theoretical section of this manual to be useful, the user will need the following:

- An understanding of partial differential equations.
- Senior level undergraduate course in linear algebra.
- Graduate or senior level undergraduate course in numerical methods.
- First level undergraduate course in groundwater flow and transport.

In order to run the code and the associate pre- and postprocessors, an understanding of the CAMCOM environment is required.

To apply SECOTP2D effectively, the user should be aware of the code's capabilities and limitations. It is recommended that the user run some of the problems provided to gain understanding in using the code.

2 GOVERNING EQUATIONS

The equation for the transport of the kth radionuclide component in a porous media (N species) can be written as (for more detailed derivation of this equation see references [5, 6, 7, 8])

$$\phi R_k\left(\frac{\partial C_k}{\partial t}\right)-\nabla\cdot\left[\phi\mathbf{D}\nabla C_k-\mathbf{V}C_k\right]=-\phi R_k\lambda_k C_k+\phi R_{k-1}\lambda_{k-1}C_{k-1}+Q\tilde{C}_k+\Gamma_k \tag{1}$$

where $k = 1,...,N$ and each dependent variable C_k (M/L^3) is the concentration of the kth radionuclide. For $k = 1$, the term involving C_{k-1} is omitted. Physical parameters include $\mathbf{D}$ (L^2/T), a 2 × 2 hydrodynamic dispersion tensor; $\mathbf{V}$ (L/T), the specific discharge; ϕ (1), the effective porosity defined as the ratio of fracture pore volume per unit volume of porous medium; R_k (1), the retardation coefficient; λ_k ($1/T$), the species decay constant; $\tilde{C}_k$ (M/L^3), the concentration of the kth injected radionuclide; Q ($1/T$), the well specific injection rate defined as the volume of liquid injected per unit volume of porous medium; and Γ_k (M/TL^3), the mass transfer term between the fracture and the matrix. The hydrodynamic dispersion tensor [6] is defined as

$$\phi\mathbf{D}=\frac{1}{|\mathbf{V}|}\begin{pmatrix}u & -v\\ v & u\end{pmatrix}\begin{pmatrix}\alpha_L & 0\\ 0 & \alpha_T\end{pmatrix}\begin{pmatrix}u & v\\ -v & u\end{pmatrix}+\phi D_k^*\tau \tag{2}$$

where α_L and α_T are the longitudinal and transverse dispersivities (L), D_k^* is the free water molecular diffusion coefficient (L^2/T), τ is the tortuosity (1), and u and v are the components of the Darcy velocity or specific discharge (L/T).

The N fracture equations are linear and sequentially-coupled. Three different boundary conditions, Dirichlet, Neumann, and flux, are available which can be used on different parts of the rectangular domain.

The flow field $\mathbf{V}$ is assumed to be independent of the solute concentration. In practice, the flow-field is obtained from the SECOFL2D code [1].

Since the dual-continuum model [6, 9] includes the exchange of mass between the matrix block and the fracture, it is necessary to solve a transport equation in the matrix block. Assuming that there is no fluid flow (advection) in the matrix, the equation for the concentration of the kth species is given by [5].

$$\phi' R_k'\left(\frac{\partial C_k'}{\partial t}\right)-\frac{\partial}{\partial\chi}\left(\phi' D'\frac{\partial C_k'}{\partial\chi}\right)=-\phi' R_k'\lambda_k C_k'+\phi' R_{k-1}'\lambda_{k-1}C_{k-1}' \tag{3}$$

where χ is the coordinate originating from the symmetry line of the matrix block, the prime denotes matrix properties, $D' = D_k^*\tau'$ is the coefficient of the molecular diffusion where τ' is the matrix tortuosity. The remaining symbols have the same meaning as those in Eq. 1. All variables in Eq. 3 are spatially dependent which gives the needed flexibility to model a clay lining.

The equations for the fracture and the matrix block are coupled through a mass transfer term Γ_k which, for a slab block model (Figure 1), is given by

$$\Gamma_k = -\frac{2}{b}\left(\phi\phi' D' \frac{\partial C_k'}{\partial \chi}\bigg|_{\chi=\frac{b'}{2}}\right) \tag{4}$$

where χ is a coordinate system originating from the center (symmetry line) of the matrix block, b is the fracture aperture, b' is the matrix block length, and ϕ is the fracture porosity defined as fracture volume per unit volume of porous medium.

For a typical matrix slab, the initial and boundary conditions are given by

$$C_k'(\chi, t=0) = C_k'^0 \tag{5}$$

$$D' \frac{\partial C_k'}{\partial \chi}(0,t) = 0 \tag{6}$$

$$C_k'\left(\frac{b'}{2}, t\right) = C_k - \zeta D' \frac{\partial C_k'}{\partial \chi} \tag{7}$$

where ζ is a parameter characterizing the resistance of a thin skin adjacent to the fracture. This parameter is defined as $\zeta = b_s/D_s$, where b_s and D_s are the skin thickness and the skin diffusion coefficient, respectively.

3 NUMERICAL DISCRETIZATION

3.1 Finite Volume Grid

SECOTP2D uses a finite volume staggered mesh as shown in Figure 2. The velocity components (u,v) are defined at cell faces and concentrations are evaluated at cell centers. The boundary conditions are applied using ghost cells. These additional cells are necessary to construct boundary conditions since there is no concentration information on cell boundaries.

3.2 Coordinate Transformation

The governing equations are transformed from Cartesian coordinates (physical space) to stretched Cartesian coordinates (computational space). The transformations are chosen so that the grid spacing in the computational space is uniform and of unit length. This produces a computational space which is a rectangular domain with a uniform mesh. The standard unweighted differencing scheme can now be applied in the numerical formulation. The metrics of the transformation are space dependent. To transform the governing equation from physical to computational space, chain rule expansions are used to represent the Cartesian derivatives in the computational space. The transformed equations, through some algebraic manipulation, are written in strong conservation form. For additional information on coordinate transformation, see Ref. [10].

3.3 Invariants of the Transformation

In the process of transforming the governing equations, additional terms containing the derivatives of the transformation are generated. These terms are collected at the end of the transformation and are known as the invariant of transformation. With the use of metric definitions, it can be shown that the invariants are analytically zero. There is an important problem associated with these invariants when equations are evaluated for uniform properties. If we require that the transformed governing equations satisfy the uniform conditions, the discretized form of the invariants must sum to zero. In 2-D, standard central differencing does satisfy this requirement.

3.4 Fracture Equation

Eq. 1 has been transformed into computational space using

$$x = x(\xi) \tag{8}$$

$$y = y(\eta) \tag{9}$$

where transformation metrics are $\xi_x = Jy_\eta$, $\eta_y = Jx_\xi$, and $J = \xi_x\eta_y$. The transformed equation, with further algebraic manipulations, was put into a strong conservation form. This is done to ensure mass conservation, which is essential here. The transformed equation is given by

$$\phi R_k \frac{\partial}{\partial t}\left(\hat{C}_k\right)+\frac{\partial}{\partial \xi}\left(\hat{E}\right)+\frac{\partial}{\partial \eta}\left(\hat{F}\right)=\frac{\partial}{\partial \xi}\left(\hat{E}_{v1}\right)+\frac{\partial}{\partial \xi}\left(\hat{E}_{v2}\right)$$
$$+\frac{\partial}{\partial \eta}\left(\hat{F}_{v1}\right)+\frac{\partial}{\partial \eta}\left(\hat{F}_{v2}\right) \tag{10}$$
$$+\phi R_k \lambda_k \hat{C}_k+\phi R_{k-1}\lambda_{k-1}\hat{C}_{k-1}$$
$$+\hat{Q}+\hat{\Gamma}$$

where

$$\hat{C}_k=\frac{C_k}{J} \tag{11}$$

$$\hat{E}=\xi_x u \hat{C}_k \tag{12}$$

$$\hat{F}=\eta_y v \hat{C}_k \tag{13}$$

$$\hat{E}_{v1}=\frac{\xi_x^2 D_{11}}{J}\frac{\partial \hat{C}_k}{\partial \xi} \tag{14}$$

$$\hat{E}_{v2}=\frac{\xi_x \eta_y D_{12}}{J}\frac{\partial \hat{C}_k}{\partial \eta} \tag{15}$$

$$\hat{F}_{v1}=\frac{\xi_x \eta_y D_{21}}{J}\frac{\partial \hat{C}_k}{\partial \xi} \tag{16}$$

$$\hat{F}_{v2}=\frac{\eta_y^2 D_{22}}{J}\frac{\partial \hat{C}_k}{\partial \eta} \tag{17}$$

$$\hat{Q}=\frac{Q\tilde{C}_k}{J} \tag{18}$$

$$\hat{\Gamma}=\frac{\Gamma}{J} \tag{19}$$

Eq. 10 is solved using an implicit Approximate Factorization procedure [10]. The advective terms are modeled by TVD [11] and the remaining terms by central differencing. A generalized three-level implicit finite volume scheme, in delta form [10], can be written as

$$\phi R_k \Delta \hat{C}_k^n=\frac{\theta \Delta t}{1+\varphi}\left(\phi R_k \Delta \hat{C}_k^n\right)_t+\frac{\Delta t}{1+\varphi}\left(\phi R_k \hat{C}_k^n\right)_t+\frac{\varphi}{1+\varphi}\left(\phi R_k \Delta \hat{C}_k^{n-1}\right) \tag{20}$$

where

$$\Delta \hat{C}_k^n=\hat{C}_k^{n+1}-\hat{C}_k^n$$

Table 1: Partial list of schemes available

θ	φ	Schemes	Truncation error
1	0	Euler, implicit	O (Δt)
½	0	Trapezoidal, implicit	O (Δt^2)
1	½	3-point-backward, implicit	O (Δt^2)

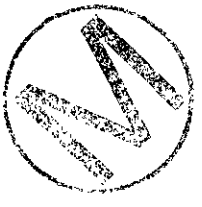
The $\Delta\hat{C}_k^n$ can be thought of as a correction to advance the solution to a new time-level (n+1). Eq. 20, with appropriate choice of the parameters θ and φ, produces many two and three-level implicit schemes as shown in Table 1. Applying Eq. 20 to Eq. 10, we have

$$\begin{aligned}
\phi R_k \Delta\hat{C}_k^n = {} & \frac{\theta\Delta t}{1+\varphi}\left[-\left(\Delta\hat{E}^n\right)_\xi - \left(\Delta\hat{F}^n\right)_n + \left(\Delta\hat{E}_{v1}^n\right)_\xi + \left(\Delta\hat{F}_{v2}^n\right)_\eta - \phi R_k\lambda_k\Delta\hat{C}_k^n + \Delta\hat{\Gamma}_k^n + \Delta\hat{Q}_k^n\right] \\
& + \frac{\theta\Delta t}{1+\varphi}\left[\left(\Delta\hat{E}_{v2}^{n-1}\right)_\xi + \left(\Delta\hat{F}_{v1}^{n-1}\right)_\eta\right] \\
& + \frac{\Delta t}{1+\varphi}\left[-\hat{E}_\xi^n - \hat{F}_\eta^n + \left(\hat{E}_{v1}^n\right)_\xi + \left(\hat{E}_{v2}^n\right)_\xi + \left(\hat{F}_{v1}^n\right)_\eta + \left(\hat{F}_{v2}^n\right)_\eta\right. \\
& \qquad \left. -\phi R_k\lambda_k\hat{C}_k^n + \phi R_{k-1}\lambda_{k-1}\hat{C}_{k-1}^n + \hat{Q}_k^n + \hat{\Gamma}_k^n\right] \\
& + \frac{\varphi}{1+\varphi}\left[\phi R_k\Delta\hat{C}_k^{n-1}\right]
\end{aligned} \tag{21}$$

or

$$\begin{aligned}
& \phi R_k\Delta\hat{C}_k^n + \frac{\theta\Delta t}{1+\varphi}\left[\left(\Delta\hat{E}^n\right)_\xi + \left(\Delta\hat{F}^n\right)_n - \left(\Delta\hat{E}_{v1}^n\right)_\xi - \left(\Delta\hat{F}_{v2}^n\right)_\eta + \phi R_k\lambda_k\Delta\hat{C}_k^n - \Delta\hat{\Gamma}_k^n - \Delta\hat{Q}_k^n\right] \\
& = \frac{\theta\Delta t}{1+\varphi}\left[\left(\Delta\hat{E}_{v2}^{n-1}\right)_\xi + \left(\Delta\hat{F}_{v1}^{n-1}\right)_\eta\right] \\
& + \frac{\Delta t}{1+\varphi}\left[-\hat{E}_\xi^n - \hat{F}_\eta^n + \left(\hat{E}_{v1}^n\right)_\xi + \left(\hat{E}_{v2}^n\right)_\xi + \left(\hat{F}_{v1}^n\right)_\eta + \left(\hat{F}_{v2}^n\right)_\eta\right. \\
& \left. -\phi R_k\lambda_k\hat{C}_k^n + \phi R_{k-1}\lambda_{k-1}\hat{C}_{k-1}^n + \hat{Q}^n + \hat{\Gamma}^n\right] \\
& + \frac{\varphi}{1+\varphi}\left[\phi R_k\Delta\hat{C}_k^{n-1}\right] \\
& = RHS
\end{aligned} \tag{22}$$

The cross derivative terms are time-lagged (n–1) to facilitate the factorization of the right-hand side operator. The error introduced by lagging these terms can be corrected through an intra-time step iteration.



The advective terms are modeled using the following TVD flux which we have developed [3] for staggered meshes. The flux is a combination of centered and locally upstream (or "upwind") weighted schemes. The basic concept of flux limiters (TVD and other algorithms) is to apply the upstream weighted scheme only locally and with only enough weighting to eliminate the non-physical oscillations which would be caused by centered differences. As the mesh is refined, the upstream weighting decreases, and the method is formally second-order accurate.

Physical space

$$\begin{aligned}\hat{E}^n_{j-\frac{1}{2},k} = \frac{1}{2}\left(1-\Phi_{j-\frac{1}{2},k}\right)\left[\left(C^n_{j,k}+C^n_{j-1,k}\right)u^n_{j-\frac{1}{2},k} - \left(C^n_{j,k}-C^n_{j-1,k}\right)\left|u^n_{j-\frac{1}{2},k}\right|\right] \\ +\frac{1}{2}\Phi_{j-\frac{1}{2},k}\left(C^n_{j,k}+C^n_{j-1,k}\right)\left(\tilde{\xi}^n_x\right)_{j-\frac{1}{2},k}u^n_{j-\frac{1}{2},k}\end{aligned} \tag{23}$$

where

$$\left(\tilde{\xi}^n_x\right)_{j-\frac{1}{2},k} = \frac{2(\xi_x)_{j,k}(\xi_x)_{j-1,k}}{(\xi_x)_{j,k}(\xi_x)_{j-1,k}}$$

Computational space

$$\begin{aligned}\hat{E}^n_{j-\frac{1}{2},k} = \frac{1}{2}a^n_{j,k}\left\{\left(1-\Phi_{j-\frac{1}{2},k}\right)\left[\left(\hat{C}^n_{j,k}+\hat{C}^n_{j-1,k}\right)-\sigma\left(\hat{C}^n_{j,k}-\hat{C}^n_{j-1,k}\right)\right]\right. \\ \left.+\frac{1}{2}\Phi_{j-\frac{1}{2},k}\left(\hat{C}^n_{j,k}+\hat{C}^n_{j-1,k}\right)\right\}\end{aligned} \tag{24}$$

where $a = \xi_x u$ and σ = sign (a).

The function Φ is called a limiter function [11]. SECOTP2D uses nine different limiter functions, six of which are TVD. The TVD limiters range from very compressive (Roe superbee) to very dissipative (minmod) [11]. The limiter functions are defined as:

0. $\Phi(r) = 0$ upstream differencing
1. $\Phi(r) = 1$ central differencing
2. $\Phi(r) = w$ weighted upstream and central differencing
3. $\Phi(r) = minmod\ (1, r)$
4. $\Phi(r) = minmod\ (1, 2\ r)$

5. $\Phi(r) = minmod\ (2, r)$

6. $\Phi(r) = minmod\ (2, 2\,r)$

7. $\Phi(r) = max\ (0, min(2, 2r), (1 + r)/2)$ van Leer's MUSCL

8. $\Phi(r) = max\ (0, min(2r, 1), min\ (r, 2))$ Roe superbee

where r is defined as

$$r = \frac{\xi_x C_\xi^n\big|_{i-\sigma+1/2}}{\xi_x C_\xi^n\big|_{i+1/2}} \tag{25}$$

$\sigma = sign\ (a_{i+1/2})$ and a is a wave speed. The minmod function is given by

$$minmod(a, b) = sign(a)\ max\ (0, min\ (|a|, sign(a) \cdot b)) \tag{26}$$

The SECOTP2D default limiter is van Leer's MUSCL.

The right hand side (RHS) of Eq. 22 involves terms that are at time level (n, n–1). After evaluation of these terms, the solution at the new time level (n+1) is obtained by

$$\left(I + \alpha_\xi L_{\xi\xi} + \alpha_\eta L_{\eta\eta} + S\right)\Delta\hat{C} = RHS \tag{27}$$

where I is an identity matrix, $L_{\xi\xi}$ and $L_{\eta\eta}$ are the x and y operators, and S is a source term. To solve Eq. 27 for $\Delta\hat{C}$, a two-dimensional banded matrix inversion is required. The banded LU solvers are adequate for small size problems; however, for large problems they require large amount of memory and CPU time. To alleviate these shortcomings, the two-dimensional operator can be Approximately Factored to represent two one-dimensional operators. This decreases the operation count for the inversion of the coefficient matrix and reduces the memory usage. The implementation of implicit boundary conditions are quite different for the factored and unfactored operators. In the case of a full two-dimensional operator, the implicit boundary conditions are applied in a straight-forward manner with no complications; however, this is not the case for the factored operators in which intermediate boundary conditions are needed to construct the implicit boundary conditions. The Approximate Factorization technique has the advantage of reducing the operation count and the memory usage for inverting the coefficient matrix, but it introduces severe complications in applying the implicit boundary conditions. With one-dimensional operators, the solution at the new time level (n+1) is obtained through the following sequence

$$\left(I + \alpha_\xi L_{\xi\xi} + a_\xi S\right)\Delta\bar{C}_n = RHS \tag{28}$$

$$\left(I + \alpha_\eta L_{\eta\eta} + a_\eta S\right)\Delta\hat{C}_n = \Delta\bar{C}_n \tag{29}$$

$$\hat{C}_{j,k}^{n+1} = \hat{C}_{j,k}^{n} + \Delta\hat{C}_{j,k}^{n} \tag{30}$$

where a_ξ and a_η are two constants that establish what percent of the source is used in each of the one-dimensional sweeps. The sum of these two constants must be 1, $a_\xi + a_\eta = 1$. Eq. 28 is the initial sweep in x or logical ξ direction. At the end of this step we have an intermediate solution stored in $\Delta\overline{C}$ which is going to be used as the right hand side to the next sweep. Eq. 29 is the sweep in y or logical η direction. At the end of this sweep the changes in the concentration $\Delta\hat{C}$ are computed and we are ready to update the magnitude of concentration by Eq. 30. The order of the sweep can be symmetrized by alternating the direction.

The boundary conditions are all implicitly implemented in the 1-D operator in both directions [10]. This ensures the second-order accuracy of the scheme. The implicit construction of boundary conditions requires an intermediate boundary condition for the initial sweep. The intermediate boundary condition is subtle and is evaluated by applying either the x or y operator to the equation of the ghost cell. The stencils of these operators will be different near the boundaries.

The following is an example of a procedure that is used to construct an intermediate boundary condition for x-sweep on the left boundary. The factored scheme for an interior point is given by

$$\left(1+\alpha_\xi L_{\xi\xi}+a_\xi S\right)\left(1+\alpha_\eta L_{\eta\eta}+a_\eta S\right)\Delta\hat{C}_{j,k}=RHS \tag{31}$$

Let ΔC^* be the intermediate boundary condition that is defined as

$$\Delta C^*_{j,k}=\left(1+\alpha_\eta L_{\eta\eta}+a_\eta S\right)\Delta\hat{C}_{j,k} \tag{32}$$

The boundary equation for a ghost cell using Dirichlet, Neumann, and flux is put into a delta form as

$$a\Delta\hat{C}_{j,k}=b\Delta\hat{C}_{j+1,k}+c \tag{33}$$

where a, b, and c are constants. Substituting Eq. 33 into Eq. 32 we have

$$\Delta C^*_{j,k}=\left(1+\alpha_\eta L_{\eta\eta}+a_\eta S\right)\left[\frac{b}{a}\Delta\hat{C}_{j+1,k}+\frac{c}{a}\right] \tag{34}$$

Simplifying

$$\Delta C^*_{j,k}=\frac{1}{a}\left[b\left(1+\alpha_\eta L_{\eta\eta}+a_\eta S\right)\Delta\hat{C}_{j+1,k}+\left(1+\alpha_\eta L_{\eta\eta}+a_\eta S\right)c\right] \tag{35}$$

Using Eq. 32 we have

$$\Delta C^*_{j,k}=\frac{1}{a}\left[b\Delta\hat{C}^*_{j+1,k}+\left(1+\alpha_\eta L_{\eta\eta}+a_\eta S\right)c\right] \tag{36}$$

At this point, Eq. 36 can be substituted into Eq. 31 and the boundary definition for the left boundary is complete. The right boundary for the *x*-sweep can be obtained in a similar manner.

For the y-sweep a general ghost cell boundary equation (bottom boundary) is written as

$$d\Delta\hat{C}_{j,k} = e\Delta\hat{C}_{j,k+1} + f \tag{37}$$

where *d*, *e*, and *f* are constants. This information is sufficient to update boundary stencils for this sweep.

Using the above procedure, the boundary information is added to the implicit part of each sweep (coefficient matrix) and then the inversion process is carried out. It should be noted that with this formulation there is no restriction on what type of boundary condition is used and the type may change from cell to cell.

3.5 Matrix Block Equation

Using a similar procedure outlined for the fracture Eq. 1, the matrix block Eq. 3 is first mapped to a computational space.

$$\phi' R_k' \left(\frac{\partial \hat{C}_k'}{\partial t}\right) - \left(\frac{\partial \hat{F}_v'}{\partial \xi'}\right) = -\phi' R_k' \lambda_k \hat{C}_k' + \phi' R_{k-1}' \lambda_{k-1} \hat{C}_{k-1}' \tag{38}$$

where

$$\hat{C}_k' = \frac{C_k'}{J} \tag{39}$$

$$\hat{F}_v' = D'\xi_x' \frac{\partial C_k'}{\partial \xi'} \tag{40}$$

Then, Eq. 38 is discretized using the general implicit finite volume scheme, in a delta form given by Eq. 20.

$$\begin{aligned} \phi' R_k' \Delta\hat{C}_k'^n = {} & \frac{\theta\Delta t}{1+\varphi}\left[\left(\Delta\hat{F}_v'^n\right)_{\xi'} - \phi' R_k' \lambda_k \hat{C}_k'^n\right] \\ & + \frac{\Delta t}{1+\varphi}\left[\left(\hat{F}'^n\right)_{\xi'} - \phi' R_k' \lambda_k \hat{C}_k'^n + \phi' R_{k-1}' \lambda_{k-1} \hat{C}_{k-1}'^n\right] \\ & + \frac{\varphi}{1+\varphi}\left[\phi' R_k' \Delta\hat{C}_k'^{n-1}\right] \end{aligned} \tag{41}$$

where

$$\left(\hat{F}_v^{'n}\right)_{j-\frac{1}{2}} = D'_{j-\frac{1}{2}}\left(\xi'_x\right)_{j-\frac{1}{2}}\left(C'^n_j - C'^n_{j-1}\right) \tag{42}$$

$$\left(\Delta\hat{F}_v^{'n}\right)_{j-\frac{1}{2}} = D'_{j-\frac{1}{2}}\left(\xi'_x\right)_{j-\frac{1}{2}}\left[J_j\Delta\hat{C}'^n_j - J_{j-1}\Delta\hat{C}'^n_{j-1}\right] \tag{43}$$

Eq. 41 is solved using a tridiagonal inversion with implicit boundary conditions. The above procedure is second-order accurate in space and time.

3.6 Fracture-Matrix Coupling

The equations for the fracture and the matrix block are coupled through a mass transfer term Γ_k. This term is proportional to the gradient of the solute concentration in the matrix block at their interface. A simple approach to couple these equations is to time lag the Γ_k term or, in other words, treat the coupling term explicitly. Our experience with *explicit* coupling has shown that if the molecular diffusion coefficient is high, and if there exists a clay lining, or there is a fine mesh resolution at the interface, the solution for the coupled system can go unstable. To make the coupling more robust, the equations must be coupled in a fully implicit manner. A procedure outlined in [6] was adapted and redeveloped for an AF algorithm with the delta formulation and a finite volume grid. This new procedure, which is adopted here, couples the equations implicitly and has proved to be quite robust.

3.6.1 Implicit Coupling

The coupling procedure consists of three steps. **Step 1** involves writing the incremental mass transfer term $\Delta\hat{\Gamma}_k^n$ in the following form that can be inserted into the implicit part of the fracture equation

$$\Delta\hat{\Gamma}_k^n = a\Delta\hat{C}_k^n + b \tag{44}$$

Step 2 involves the evaluation of a and b terms. This is accomplished using the inversion process (LU factorization) in the solution of the matrix equation. After the construction of the lower tridiagonal matrix L and the intermediate solution, there is enough information to evaluate the Δ terms. This new information is fed into the fracture equation which subsequently is solved for concentrations in the fracture at the new time level (n+1). **Step 3** involves constructing the boundary condition for the matrix equation at the fracture-matrix interface using fracture concentrations at the (n+1) time level. Matrix concentrations are then obtained using the upper tridiagonal matrix U by back substitution.

3.7 Solution Accuracy

The present code uses TVD limiters with three-level time differencing and directional splitting to improve accuracy and execution time. The code is second-order accurate both in time and space with appropriate time and spatial discretizations; however, TVD limiters in the vicinity of a sharp spatial gradient such as a propagating front make the solution locally first-order accurate. The

spatial accuracy of the SECOTP2D code with the TVD option on is less than second-order accurate, and the deviation from second-order accuracy will depend on how many sharp fronts exist in the computational plane. Problems with a moderately high Peclet number would greatly benefit from the TVD scheme by avoiding spurious oscillations commonly associated with the central differencing schemes. The long time scales of the problems to which the code is to be applied dictate the use of implicit algorithms.

The flow field is usually computed by SECOFL2D, but any comparable flow code can be used. It is important to note that the convergence tolerance on the flow must be smaller in magnitude than the *source* ($\hat{Q}$) for the transport calculation. Lack of proper iterative convergence in the flow calculation can show up as a source term in the transport calculation due to its conservative formulation and in some cases can lead to instabilities.

In practice, the transport code is sensitive to a flow field which is erratic (non-smooth) at the boundaries. Any flow field that exhibits significant flow structure near a boundary suggests that the location of the boundaries are incorrect, i.e., the conceptual modeling is inadequate. We assume that if the solution on the boundaries is not known, then they should be located far enough away from any major structures in the flow field.

Lack of smoothness of the flow field near boundaries can pose difficulties for the transport calculation. The SECOTP2D code, in most cases, can eliminate these difficulties by automatically assigning the boundary conditions using the flow field. This is done by comparing the magnitude of the normal velocity component for each cell on four sides of the computational domain to a value ε which is taken to be a 0.1 % of the maximum velocity magnitude. For cells in which the normal velocity components are below ε, the boundary conditions are set to Dirichlet boundary condition with zero concentration. The boundary conditions for the remaining cells are set by examining the flow field. If the normal velocity component is outward the boundary condition is set to a Neumann condition with zero gradient and if it is inward the boundary condition is set to a Dirichlet condition with zero concentration.

4 CODE VERIFICATION

The code, which has been developed based on the scheme described in the algorithm section, is verified for temporal and spatial accuracy against analytical solutions. The single and multiple species fracture transport, matrix diffusion, and coupling procedure parts of the algorithm are verified individually.

The transport in the fracture for a single species contains advection, diffusion, and decay; therefore, the benchmark should exercise these components of the numerical algorithm. Benchmarks (I) and (II) are given by Knupp (see Appendix). Benchmark (I) includes a time dependent spatially variable velocity field and a spatially variable dispersivity field; however, the boundary conditions are not time dependent. Benchmark (II) is used to verify the implicit boundary condition of the code within the operator splitting and the Approximate Factorization scheme.

The matrix equation is verified against an analytical solution which provides spatial variability in the properties with time dependent boundary conditions.

Having verified the fracture and the matrix solution for the transport of a single species individually, the only remaining part of the numerical solution is the coupling between the two equations. This implicit coupling procedure is verified using the Tang [13] analytical solution.

For multiple species transport, the code is benchmarked against the analytical solution formulated by Lester et. al. [14] which provides the solution for three species in a chain.

SECOTP2D also provides discharge calculations on different pre-defined closed boundaries. Since the discharge calculations involve integration, this process needs to be verified. This is done using benchmark (I).

Uniform and stretched grids are used in computing benchmark cases where analytical solutions exist. Errors are estimated by comparing the computed solutions with the analytical solutions and using either an ℓ_2- or infinity-norm. The accuracy or the order of convergence of these solutions are ascertained using Roache's Grid Convergence Index GCI [15].

Following this section on code verification, further confirmation is presented in Section 5 on the Sudicky-Frind problem.

4.1 Error Estimator and GCI

The Grid Convergence Index GCI [15] provides an objective approach to the evaluation of uncertainty in grid convergence studies. The GCI is constructed from a grid refinement error estimator which is derived from the theory of generalized Richardson extrapolation. The GCI is a numerical error band equal to 3 times the error estimate. Grid doubling is not a requirement for GCI. This flexibility is an important asset for problems in which grid doubling is not feasible. The GCI is defined for the fine grid (in a coarse-fine pair) as

$$GCI\,(\textit{fine grid}) = \frac{3|\varepsilon|}{r^p - 1} \tag{45}$$

where ε is defined as

$$\varepsilon = \frac{f_{coarse} - f_{fine}}{f_{fine}} \tag{46}$$

f_{coarse} and f_{fine} are coarse and fine grid solutions, $r > 1$ is the grid refinement ratio, and p is the order of the method.

The theoretical or expected order p can be verified experimentally by examining the ratio of GCIs. For example, two GCIs can be computed with three grid solutions, from the fine grid to the intermediate grid (GCI_{12}), and from the intermediate to the coarse grid (GCI_{23}). The theory predicts that

$$r^p = \frac{GCI_{23}}{GCI_{12}} \tag{47}$$

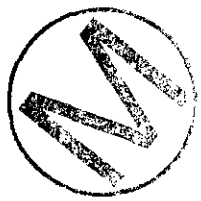
If the results from a grid refinement (with constant r) approximately satisfy this relation, it (a) verifies the order p, and (b) indicates that the asymptotic range is achieved. For additional information on how GCI is computed, see Refs. [15, 16, 17].

4.2 Fracture Equation

The transport in the fracture contains advection, diffusion, and decay; therefore, the benchmark should exercise these components of the numerical algorithm. An analytical solution to unsteady fracture transport is given by Knupp (see Appendix) that includes a time dependent, spatially variable velocity field, and a spatially variable dispersivity field. The boundary conditions are time independent and have zero values on all boundaries. The analytical solution assumes zero free-water molecular diffusion, no dual porosity term, and single-species decay. The benchmark tests are organized as follows.

1. Steady State Calculations

   - time-independent boundary conditions
   - uniform and stretched grid
   - central differencing
   - TVD limiters
     - minmod(1, r)
     - minmod(1, 2r)
     - minmod(2, r)
     - minmod(2, 2r)
     - van Leer's MUSCL
     - Roe superbee



2. Unsteady Calculations

   - 3-point backward and trapezoidal time differencing
   - time-independent boundary conditions

- central differencing and van Leer MUSCL limiter
- uniform and stretched grid

3. Implicit Boundary Condition Verifications

- time-dependent boundary conditions
- central differencing
- uniform grid

4.2.1 Steady State Calculations

For the steady state calculations, parameters are: $0 \le x \le 1$ m, $0 \le y \le 1$ m, $\phi = 1.0$, R = 1.0, λ = 0.01/day, $\omega_1 = 0.0$, $\omega_2 = 0.0$, $u_o = 0.02$ m/day, $v_o = 0.01$ m/day, $\alpha_{Lo} = 10.0$ m, and $\alpha_{To} = 1.0$ m. For additional information and definition of variables, see Appendix. All the steady state computations are converged to machine zero. There were five grid sizes used in the convergence test and they are all related by grid doubling (r = 2): 10×10, 20×20, 40×40, 80×80, and 160×160.

Case 1.1 - Uniform grid and TVD limiter = 1, central differencing. Table 2 presents the results for the convergence test. The ratios of GCIs in column 5 have remained relatively constant and equal to 4; hence, the solutions are second order accurate ($r = 2$, $r^p = 4$, so $p = 2$).

Table 2: Benchmark (I): steady state calculations, TVD limiter = 1; central differencing, uniform grid.

Grid	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	4.0507E-3	7.5701E-3		
20×20	9.7919E-4	1.9282E-3	202.673 %	
40×40	2.4268E-4	4.8200E-4	48.599 %	4.17
80×80	6.0559E-5	1.2054E-4	12.017 %	4.05
160×160	1.5155E-5	3.0171E-5	2.996 %	4.01

Case 1.2 - Uniform grid and TVD limiter = 3-6, minmod limiters. Tables 3-6 present the results for the convergence tests. Column 5 presents the ratio of successive GCIs. Among the minmod limiter functions, limiter number 4 which is minmod(1, 2r), shown in Table 4, has exhibited a second-order accurate solution. The remaining minmod limiters have shown an order between first and second-order accurate behavior which is consistent with the way TVD limiters are constructed. The last GCI ratio for the finest grid pair in Tables 3, 5, and 6 show a drop below 4. This behavior is often associated with computer word-length limitations.

Table 3: Benchmark (I): steady state calculations, TVD limiter = 3; minmod (1,r), uniform grid.

Grid	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	3.3663E-3	6.9064E-3		
20×20	7.3017E-4	1.5570E-3	41.997 %	
40×40	2.2055E-4	5.9687E-4	8.119 %	5.17
80×80	1.1293E-4	2.9931E-4	1.715 %	4.73
160×160	6.2770E-5	1.5754E-4	0.799 %	2.15

Table 4: Benchmark (I): steady state calculations, TVD limiter = 4; minmod (1,2r), uniform grid.

Grid	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	4.0259E-3	7.6351E-3		
20×20	9.7757E-4	1.9328E-3	210.077 %	
40×40	2.4267E-4	4.8411E-4	48.476 %	4.15
80×80	6.0588E-5	1.2079E-4	12.011 %	4.04
160×160	1.5160E-5	3.0198E-5	2.997 %	4.01

Table 5: Benchmark (I): steady state calculations, TVD limiter = 5; minmod (2,r), uniform grid.

Grid	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	3.3666E-3	6.9072E-3		
20×20	7.2917E-4	1.5517E-3	42.026 %	
40×40	2.1956E-4	5.9444E-4	8.120 %	5.17
80×80	1.1279E-4	2.9866E-4	1.701 %	4.77
160×160	6.2757E-5	1.5745E-4	0.797 %	2.13

Table 6: Benchmark (I): steady state calculations, TVD limiter = 6; minmod (2,2r), uniform grid.

Grid	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	5.3927E-3	9.7949E-3		
20×20	1.6961E-3	3.0871E-3	31.476 %	
40×40	6.2195E-4	1.2384E-3	9.146 %	3.44
80×80	2.5875E-4	5.4452E-4	3.093 %	2.96
160×160	1.1744E-4	2.5382E-4	1.203 %	2.57

Case 1.3 - Uniform grid and TVD limiter = 7, van Leer's MUSCL limiter. Table 7 presents the results for the convergence test. As the results in column 5 show, the solutions are second-order accurate. This limiter function is not too dissipative or compressive; because of this, it is chosen to be the default limiter for SECOTP2D under the TVD option.

Table 7: Benchmark (I): steady state calculations, TVD limiter = 7; van Leer's MUSCL, uniform grid.

Grid	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	4.0311E-3	7.6009E-3		
20×20	9.8145E-4	1.9341E-3	199.584 %	
40×40	2.4408E-4	4.8392E-4	48.257 %	4.14
80×80	6.1008E-5	1.2110E-4	11.981 %	4.03
160×160	1.5280E-5	3.0324E-5	2.993 %	4.00

Case 1.4 - Uniform grid and TVD limiter = 8, Roe's superbee limiter. Table 8 presents the results for the convergence test. This is a compressive limiter function and is designed to work with discontinuities in the solution. For this benchmark, which has a smooth solution, the limiter has shown second-order accurate behavior (column 5).

Table 8: Benchmark (I): steady state calculations, TVD limiter = 8; Roe's superbee, uniform grid.

Grid	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	4.2109E-3	7.7523E-3		
20×20	1.0268E-3	1.9334E-3	198.981 %	
40×40	2.5551E-4	4.8476E-4	48.200 %	4.13
80×80	6.3885E-5	1.2129E-4	11.975 %	4.03
160×160	1.6002E-5	3.0369E-5	2.992 %	4.00

Case 1.5 - Nonuniform grid and TVD limiter = 1, central differencing. Table 9 presents the results for the convergence test. The Δs corresponds to a minimum cell size. Geometric stretchings, with given initial cell sizes, are used to construct the grids. As the results in column 5 show, the solutions are second-order accurate.

Table 9: Benchmark (I): steady state calculations, TVD limiter = 1; central differencing, non-uniform grid.

Grid	Δs	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	0.04	9.5779E-3	2.7666E-2		
20×20	0.02	2.2588E-3	6.0692E-3	217.352 %	
40×40	0.01	5.4968E-4	1.4608E-3	50.755 %	4.28
80×80	0.005	1.3558E-4	3.5703E-4	12.297 %	4.13
160×160	0.0025	3.3674E-5	8.8283E-5	3.026 %	4.06

4.2.2 Unsteady Calculations

For the time-dependent calculations parameters are: ϕ = 1.0, R = 1.0, λ = 0.001 1/day, ω_1 = 0.0157, ω_2 = 0.00785, u_o = 0.02 m/day, v_o = 0.01 m/day, α_{Lo} = 10.0 m, α_{To} = 10 m, and Time = 0.8 day. The numerical errors and the order of the schemes are evaluated in the same manner as was described previously.

Case 2.1 - Trapezoidal time differencing (second-order), TVD limiter = 1, central differencing, and uniform grid. Table 10 presents the results for the convergence test.

Table 10: Benchmark (I): time-dependent calculations, trapezoidal time differencing, TVD limiter = 1; central differencing, uniform grid.

Grid	Δt	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	0.04	3.3879E-3	6.8374E-3		
20×20	0.02	8.2277E-4	1.7013E-3	207.049 %	
40×40	0.01	2.0354E-4	4.2555E-4	49.982 %	4.14
80×80	0.005	5.0385E-5	1.0567E-4	12.362 %	4.04
160×160	0.0025	1.2389E-5	2.6053E-5	3.067 %	4.03

This exercise verifies the order of the overall scheme including the time dependent part of the algorithm. As the results in column 5 show, the solutions are second-order accurate in time and space.

Case 2.2 - Trapezoidal time differencing (second-order), TVD limiter = 7, van Leer's MUSCL limiter function, and uniform grid. Table 11 presents the results for the convergence test. Again, solutions are second-order accurate as shown in column 5.

Table 11: Benchmark (I): time-dependent calculations, trapezoidal time differencing, TVD limiter = 7; van Leer's MUSCL, uniform grid.

Grid	Δt	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	0.04	3.3742E-3	6.8663E-3		
20×20	0.02	8.2240E-4	1.7046E-3	205.360 %	
40×40	0.01	2.0385E-4	4.2610E-4	49.779 %	4.13
80×80	0.005	5.0512E-5	1.0590E-4	12.340 %	4.03
160×160	0.0025	1.2426E-5	2.6112E-5	3.065 %	4.03

Case 2.3 - Trapezoidal time differencing (second-order), TVD limiter = l, central differencing, and non-uniform grid. As Table 12 shows, the results are second-order accurate.

Table 12: Benchmark (I): time-dependent calculations, trapezoidal time differencing, TVD limiter = 1; central differencing, non-uniform grid.

Grid	Δs	Δt	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	0.04	0.04	7.7854E-3	2.1118E-2		
20×20	0.02	0.02	1.8912E-3	5.0190E-3	208.290 %	
40×40	0.01	0.01	4.6400E-4	1.2186E-3	50.435 %	4.13
80×80	0.005	0.005	1.1448E-4	2.9891E-4	12.351 %	4.08
160×160	0.0025	0.0025	2.8298E-5	7.3579E-5	3.046 %	4.05

Case 2.4 - 3-point backward time differencing (second-order accurate), TVD limiter = 1, central differencing, and uniform grid. Table 13 presents the results for the convergence test. As the results show, solutions are second order accurate and are similar to that generated by the trapezoidal time differencing.

Table 13: Benchmark (I): time-dependent calculations, 3-point backward time differencing, TVD limiter = l; central differencing, non-uniform grid.

Grid	Δt	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
10×10	0.04	3.3983E-3	6.8363E-3		
20×20	0.02	8.2551E-4	1.7024E-3	203.078 %	
40×40	0.01	2.0472E-4	4.2682E-4	49.001 %	4.14
80×80	0.005	5.0952E-5	1.0642E-4	12.137 %	4.04
160×160	0.0025	1.2669E-5	2.6436E-5	3.022 %	4.02

4.2.3 Implicit Boundary Conditions

Benchmark (II), which is given by Knupp (see Appendix), is used to verify the implicit boundary conditions of the code. Since the computational domain is finite, the Dirichlet boundary conditions are time dependent and may be obtained from the exact solution.

Case 3.1: The parameters are: time = 25 days, u = 0.1 m/day, α_L = 1.0 m, and α_T = 0.1 m. Four different grid sizes and time steps are used in this convergence study. The code was set to use 3-point backward time differencing which is second-order accurate and a TVD limiter = 1 (central differencing). Table 14 presents the computed solution to Benchmark (II), the error, and the GCIs. By examining the ratio of GCIs, it is evident that the overall solution is second-order accurate in time and space. Therefore, the implicit treatment of boundary conditions in the AF algorithm is verified to be second-order accurate in time.

Table 14: Benchmark (II), time-dependent calculations, 3-point backward time differencing, TVD limiter = 1; central differencing, uniform grid.

Grid	Δx	Δt	$\ell_2 - norm$	GCI	GCI ratio (r^p)
20×20	0.05	0.25	7.697E-3		
40×40	0.025	0.125	1.954E-3	46.540 %	
80×80	0.0125	0.0625	4.921E-4	11.847 %	3.92
160×160	0.00625	0.03125	1.234E-4	2.988 %	3.96

4.3 Matrix Block Equation

The numerical algorithm to solve the one-dimensional matrix block equation, described in Section 3.5, is verified against the following analytical solution. The governing equation is given by

$$\rho\frac{\partial c}{\partial t} = \frac{\partial}{\partial x}\left(D\frac{\partial c}{\partial x}\right) - \rho\lambda c \tag{48}$$

and $0 \le x \le b$. The variables D and ρ are defined as

$$D = \sqrt{2}\alpha\left(e^x - x\right)^{\frac{1}{2}} \tag{49}$$

$$\rho = \frac{e^x}{\sqrt{2}\left(e^x - x\right)^{\frac{1}{2}}} \tag{50}$$

The boundary and initial conditions are

$$\left.\frac{\partial c}{\partial x}\right|_{x=0} = 0 \tag{51}$$

$$c(b,t) = \sqrt{2}\left(e^b - b\right)^{\frac{1}{2}} e^{(\alpha-\lambda)t} \tag{52}$$

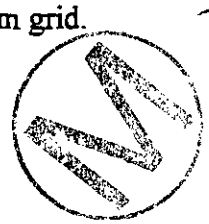
$$c(x,0) = \sqrt{2}\left(e^x - x\right)^{\frac{1}{2}} \tag{53}$$

The solution is

$$c(x,t) = \sqrt{2}\left(e^x - x\right)^{\frac{1}{2}} e^{(\alpha-\lambda)t} \tag{54}$$

The convergence tests are done for steady state calculations using uniform and nonuniform grids and for unsteady calculations with trapezoidal and 3-point backward time differencing with a uniform grid.

For steady cases, the parameters are $\alpha = \lambda = 1$ and $b = 4.0$. All the steady calculations are converged to machine zero. Table 15 presents the convergence results for a steady, uniform grid.



As the behavior of the ratio of GCIs show (column 5), the solutions are second-order accurate. Table 16 shows the steady calculations with non-uniform grid. Again, from the results in column 5, the solutions are second-order accurate.

Table 15: Matrix verification: steady state calculations, uniform grid.

Grid	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
20	6.3588E-3	1.4884E-2		
40	1.4470E-3	3.4894E-3	235.917 %	
80	3.4509E-4	8.4329E-4	52.926 %	4.46
160	8.4264E-5	2.0719E-4	12.528 %	4.22
320	2.0820E-5	5.1343E-5	3.047 %	4.11

Table 16: Matrix verification: steady state calculations, non-uniform grid.

Grid	Δs	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
20	0.04	4.7240E-3	1.1464E-2		
40	0.02	1.0251E-3	2.3426E-3	244.200 %	
80	0.01	2.4657E-4	5.6313E-4	51.398 %	4.75
160	0.005	6.0886E-5	1.3810E-4	12.259 %	4.19
320	0.0025	1.5147E-5	3.4211E-5	3.020 %	4.06

For unsteady calculations, the parameters are $\alpha = 1$, $\lambda = 5$, $b = 4.0$, and $t = 1.0$. Table 17 presents the convergence results for unsteady, trapezoidal time differencing and uniform grid. As the ratios of successive GCIs in column 5 show, the solutions are second-order accurate both in time and space. The results obtained with 3-point backward time differencing are similar to those computed by trapezoidal time differencing.

Table 17: Matrix verification: unsteady calculations, uniform grid.

Grid	Δt	$\ell_2 - norm$	Max. error	GCI	GCI ratio (r^p)
20	0.1	8.6435E-3	1.0795E-2		
40	0.05	1.8384E-3	2.3088E-3	267.433 %	
80	0.025	4.2851E-4	5.3931E-4	55.407 %	4.83
160	0.0125	1.0353E-4	1.3041E-4	12.771 %	4.34
320	0.00625	2.5446E-5	3.2062E-5	3.069 %	4.16

4.4 Coupling Procedure

To verify both the fracture and the matrix finite volume discretization as a system and the coupling procedure, we have chosen a dual porosity problem in one dimension with the analytical solution given by Tang [13]. The Tang solution is numerically evaluated and hence there is an error associated with its evaluation. This is due to accuracy of the numerical algorithm used in

generating the solution. In this case, the grid convergence test is not possible since the computed solution ("exact") is not accurate enough.

The fracture equation is

$$\frac{\partial c}{\partial t}+\frac{v}{R}\frac{\partial c}{\partial z}-\frac{D}{R}\frac{\partial^2 c}{\partial z^2}+\lambda c-\left.\frac{\theta D'}{bR}\frac{\partial c'}{\partial x}\right|_{x=b}=0 \tag{55}$$

where $0 \le z < \infty$. The fracture is along the z coordinate and the matrix block uses the x coordinate. The initial and boundary conditions are

$$c(0,t)=c_0 \tag{56}$$
$$c(\infty,t)=0 \tag{57}$$
$$c(z,0)=0 \tag{58}$$

The matrix equation is given by

$$\frac{\partial c'}{\partial t}-\frac{D'}{R'}\frac{\partial^2 c'}{\partial x^2}+\lambda c'=0 \tag{59}$$

where $b \le x < \infty$. The initial and boundary conditions are

$$c'(b,z,t)=c(z,t) \tag{60}$$
$$c'(\infty,z,t)=0 \tag{61}$$
$$c'(x,z,0)=0 \tag{62}$$

For further explanation of the problem, definition of parameters, and the analytical solution, see Ref. [13].

The test problem is set up by defining the required parameters as follows: the fracture is along the x coordinate, fracture length, L = 10 *m*, fracture spacing 2.4 *m*. *Fracture properties:* aperture, b = 1.0E-4 *m*, seepage velocity, V = 0.01 *m/d*, longitudinal dispersivity, α_L = 0.50 *m*, molecular diffusion coefficient, D = 1.382E-4 m^2/d, and fracture porosity, ϕ = 0.42E-4. *Matrix properties:* the matrix uses the χ coordinate (see Figure 1), matrix porosity, ϕ' = 0.01, and matrix diffusion coefficient, D' = 1.382E-7 m^2/d. *Radionuclide properties:* decay constant, λ = 0.154E-3 *1/d*, and retardation factor, $R = R'$ = 1. *Initial conditions:* $c(x, 0) = c'(x, \chi, 0) = 0$. The boundary conditions are

$$c(0,t)=1 \tag{63}$$
$$\frac{\partial c}{\partial x}(L,t)=0 \tag{64}$$
$$c'\left(x,\frac{b'}{2},t\right)=c(x,t) \tag{65}$$

$$\frac{\partial c'}{\partial \chi}(0,t) = 0 \qquad (66)$$

Fracture length and matrix block length are discretized using 80 and 15 stretched cells, respectively. The calculation was stopped at time = 100 days to test both spatial and temporal accuracy of the computed solution. Figures 3 and 4 present the comparison of the fracture and matrix solution to the analytical solution. The computed results reproduce the analytical solutions in both regions, which also verifies the accuracy of the coupling procedure. Further mesh refinement in both fracture and matrix block reproduced the same results.

4.5 Multiple Species

For multiple species transport, SECOTP2D is benchmarked against the Lester et. al. [14] analytical solution. This analytical solution, which includes the effect of axial dispersion, describes the transport of the radionuclide chains through a column of adsorbing media. Initially, all species have zero concentrations and the solutions of Bateman equations are used as the time dependent inlet boundary conditions. For this exercise, we are using a three-member decay chain. The problem parameters are: time = 100 yr, length = 0.1 km, v = 0.1 km/yr, α_L = 0.5 km, and the species dependent properties are given in Table 18. The computed results were obtained with 80 cells in the column, Δt of 0.25 year, and 400 steps. Figures 5 and 6 present the computed and the analytical results for all the species. As these figures show, the computed results reproduce the analytical solutions.

Table 18: Half-lives and equilibrium constants of nuclides.

Species	Half-lives (yr)	Decay Constant (l/yr)	Equilibrium Constant
1	15	0.04621	10,000
2	433	0.00160	10,000
3	6540	0.00011	10,000

4.6 Discharge Calculations

The ability of SECOTP2D to compute the discharge (mass flux) on a predefined closed boundary is verified using benchmark (I). The integral defining the discharge is given in the Appendix. The numerical quadrature used is trapezoidal which is second-order accurate. Benchmark (I) is exercised in the same manner as described for case 1.1 (uniform grid) and case 1.5 (stretched grid). A single closed boundary which is a rectangular box is specified by using the coordinates of the upper left hand corner and the lower right hand corner of the box. For the uniform grid, the coordinates of these points are (0.2,0.8) and (0.8,0.2), and for the stretched grid they are (0.2085,0.7915) and (0.7915,0.2085). Table 19 and 20 present the results for the convergence study using five different grids. As the ratio of GCIs show, the computation of discharges are second-order accurate for both uniform and stretched grid.

Table 19: Discharge calculations, uniform grid, benchmark (I)

Grid	Computed	Error	GCI	GCI ratio (r^p)
10×10	0.411362	3.4856E-3		
20×20	0.408745	8.6837E-4	193.639 %	
40×40	0.408093	2.1690E-4	48.200 %	4.02
80×80	0.407931	5.4213E-5	12.037 %	4.00
160×160	0.407890	1.3516E-5	3.011 %	4.00

Table 20: Discharge calculations, stretched grid, benchmark (I)

Grid	Δs	Computed	Error	GCI	GCI ratio (r^p)
10×10	0.04	0.364443	3.182E-3		
20×20	0.02	0.362069	8.089E-4	189.696%	
40×40	0.01	0.361464	2.033E-4	48.409%	3.92
80×80	0.005	0.361311	5.085E-5	12.186 %	3.97
160×160	0.0025	0.361273	1.251E-5	3.065 %	3.98

5 CODE CONFIRMATION PROBLEMS

In the previous section 4, we have exercised all features of the code and have verified its accuracy in all aspects. The present section gives further confirmation calculations. Even if a code is rigorously verified, in the sense of a mathematical theorem, it can still be worthwhile to present selected additional confirmation calculations for the purpose of user confidence building.

5.1 Convergence Demonstration on WIPP PA Problems

To demonstrate convergence on typical problems, we will use two of the vectors from the 1992 WIPP PA calculation [18].

5.1.1 Fracture Transport

Vector 2 (E1E2 scenario) was chosen for a grid convergence demonstration for fracture transport. This vector has moderate parameters, such as fracture aperture and fracture travel time.

Since we do not have an exact solution for vector 2, we rely on contours of the solution for judging convergence. We will use three different grid sizes, 46×53, 93×107, and 187×215. For each grid size three different time steps are used, Δt = 10, 5, and 2.5 years, to show time convergence.

Figure 7 shows temporal behavior of the source function over 10,000 years. Figures 8a, 8c, and 8e present the contours of solute concentrations on the first grid at t = 10,000 years for three different time steps, respectively. The time resolution for this mesh is quite adequate since there is only slight change between contour plots. Figures 8b, 8d, and 8f present breakthrough curves, with each plot presenting integrated discharges through three closed boundaries. The breakthrough curves also show convergence in time. Figures 9 and 10 show similar plots for grids number 2 and 3. As we refine the grid, the plume becomes narrower and the concentration front becomes sharper. This is due to improved effectiveness of the TVD algorithm.

These sequences of grid and time steps clearly show that the problem is not adequately resolved on the coarse grid.

5.1.2 Dual-Porosity Transport

For a dual-porosity transport calculation, vector 52 (E1E2 scenario) is a typical example which has no extremes in its parameters. We will use the same grid sizes as in the fracture transport case; however, vector 52 has different time scales for both the fracture and the matrix block and requires different time steps, Δt = 2, 1, and 0.66 years.

Figure 11 shows temporal behavior of the source function over 10,000 years. Figures 12a, 12c, and 12e present the solute concentration on the first grid at time = 10,000 years for the three different time steps, respectively. Similar to the fracture calculation, the time resolution is satisfactory. Figures 12b, 12d, and 12f present breakthrough curves. Again, there are no significant changes in the solution (visual detection) as the Δt decreases. Figures 13 and 14 show a similar plot for grids 2 and 3. As we observed in the fracture calculation, the concentration

front becomes sharper as the grid becomes finer. Figure 12c shows some discharge on the side boundary whereas on the finer meshes there are no discharges. This points out that the first grid is not resolving the solution adequately; however, the other grids are adequate.

5.2 Dual-Porosity Solution of Sudicky and Frind

For dual-porosity calculations, SECOTP2D was benchmarked using Tang's [13] analytical solution. This solution has a limitation that the matrix block length is infinite. Sudicky and Frind published an analytical solution in 1982 [19] for contaminant transport in a fractured porous media that removed this limitation. The original Sudicky-Frind analytical solution had some errors which were corrected by Davis and Johnston [20]. With this solution SECOTP2D can be exercised for a finite matrix block length in the dual porosity calculation. Van Gulick [21] has numerically implemented the Sudicky-Frind analytical solution. He has performed a thorough evaluation of the accuracy of the numerical solution and the range of parameters in which the solution converges. Herein, the data generated by van Gulick are used to compare with the unsteady and steady dual porosity calculations.

The following problem has been set up from the data provided by van Gulick. The problem is a one-dimensional longitudinal transport in the fracture with transverse matrix diffusion. For the unsteady calculation, the solution is evaluated at 500 days.

Fracture properties: aperture, b = 0.1E-3 (m); ϕ_f = 1.9996E–4, longitudinal dispersivity, α_L = 0.1 (m); molecular diffusion coefficient, D_f = 1.3824E-4 (m^2/day); pore velocity V = 0.0075 (m/day); decay constant, λ = 1.5366E-3 ($1/day$); fracture retardation, R = 1.0; fracture length, L = 2.0 (m).

Matrix properties: block length, b' = 0.50 (m); matrix porosity, ϕ_m = 0.01; diffusion coefficient, D' = 1.3824E-7 (m^2/day); retardation, R= 1.0.

The fracture and the matrix concentrations are set to zero as the initial condition. The boundary conditions for the fracture are

$$C(0,t) = 1.0 \tag{67}$$

$$\frac{\partial C(L,t)}{\partial x} = 0 \tag{68}$$

where $0 \le x \le L$.

The fracture was discretized with 200 uniform cells based on the grid convergence study for the single porosity calculations. The matrix was discretized with 5 to 80 uniform cells. Figure 15 presents the grid convergence test for the matrix and the comparison to the analytical solution. The final solution was computed using 200 uniform cells in the fracture and 80 uniform cells in the matrix. For the unsteady calculation, based on the convergence study, the time step was set to 1 day. The solution was converged to machine zero for steady state calculation.

Figure 16 presents the concentration profile in the fracture compared to the Sudicky-Frind analytical solution at 500 days; the agreement is excellent. Figure 17 shows the comparison of the concentration profiles in the matrix, at three different locations in the fracture, compared to the analytical solutions. Again, the agreement is excellent. Figure 18 presents the concentration

profile in the fracture at steady state compared to the analytical solution. Again, we have excellent agreement.

5.3 High Peclet Transport Case with TVD

To illustrate the advantages of the TVD algorithm presented in Chapter 3, we have chosen to solve a two-dimensional convection-dispersion problem for which we have an exact solution [22]. The medium is assumed to be homogeneous and isotropic with a unidirectional steady-state flow. The initial solute concentration is zero. At $t = 0$ a strip-type source with a finite length 2a, as shown in Figure 19, along the y-axis is introduced. The source concentration remains constant with time. For additional information, see Ref. [22].

The solution is obtained for two cases, low and high mesh Peclet numbers, where mesh Peclet number is defined as $\Delta x/\alpha_L$. For both cases, a uniform 80×80 grid with $0 \le x \le 100$ m and $-50 \le y \le 50$ m is used. The code was set up to use the TVD, van Leer's MUSCL limiter, and trapezoidal time differencing.

Low mesh Peclet number case: Pe = 2, u = 1.0 m/day, v = 0.0 m/day, α_L = 0.625 m, α_T = 0.0625 m, λ = 0.0, a = 25 m, and time = 30 days. Figure 20 presents the analytical solution. Figure 21 presents the computed solution (Δt = 0.1 day) using TVD, van Leer's MUSCL limiter, and Figure 22 presents the computed solution using upstream differencing. It is clear from these figures that even at a low mesh Peclet number the upstream solution, due to its artificial diffusion, is not accurate; however, the TVD solution is very close to the analytical solution.

High mesh Peclet number case: Pe = 10, u = 1.0 m/day, v = 0.0 m/day, α_L = 0.125 m, α_T = 0.0125 m, a = 25 m, and time = 30 days. Figure 23 shows results from the analytical solution. Figure 24 presents the TVD solution and Figure 25 presents the upstream solution (Δt = 0.1 day). The difference between these two solutions is dramatic. As expected, the TVD scheme retained a sharp front; close to the analytical solution, as opposed to a very diffused front generated by the implicit upstream differencing.

As shown above, the TVD scheme in conjunction with second-order time discretization is more accurate in tracking sharp fronts than the classical upstream schemes, even for low mesh Peclet number cases.

6 HARDWARE PLATFORMS

The portability of codes is always a question; therefore, it was essential to verify that the computed results from SECOTP2D are not machine sensitive. The code was run, using benchmark (I) (case 1.1), on the following platforms: Digital Alpha, HP 9000/720, CRAY Y-MP C90, Silicon Graphics IRIS 4D-25, Sun 4c Spark station, and Micro-Vax II. The CRAY code is single precision while the others are double precision. For simplicity, a coarse grid of 10×10 and 20 time steps were used. The -norm of the error, which is computed from the contributions of all the computational cells, and the maximum error are good values to compare on different platforms. Initially, machine zeros were evaluated for all the computers. Table 21 presents the machine type, operating systems, and machine zeros. It is interesting to note that machine-zero were-the same for all computers (double precision) except for the CRAY (single precision) which was about two orders of magnitude larger than the other machines. Table 22 presents the machine type, -norm of the error, and the maximum error (benchmark (I), case 1.1). As this table shows, all the computed results are the same within the prescribed precision of each machine. This clearly indicates that the results generated by the SECOTP2D code are not machine sensitive.

Table 21: Hardware, operating systems, and machine-zeros

Hardware	Operating System	Machine-Zero
Alpha	OpenVMS 6.1	1.110E-16
HP	HP-UX A.09.01, (unix)	1.110E-16
CRAY (SP)	UNICOS	7.105E-15
IRIS	IRIX 4.0.5, (unix)	1.110E-16
Sun	Sun OS 4.1.1, (unix)	1.110E-16
Micro-VAX II	VMS V5.2	1.110E-16

Table 22: Computed results on different platforms benchmark (I)

Hardware		Max. error
Alpha	3.910447413657691E-03	7.464205046390449E-03
HP	3.910447413657638E-03	7.464205046390337E-03
CRAY (SP)	3.910447413643900E-03	7.464205046375100E-03
IRIS	3.910447413657621E-03	7.464205046390448E-03
Sun	3.910447413657600E-03	7.464205046390300E-03
Micro-VAX II	3.910447413657696E-03	7.464205046390449E-03

7 USER INTERACTIONS, INPUT AND OUTPUT FILES

In order to run SECOTP2D, a preprocessor, PRESECOTP2D must first be run to setup all of the input files that are needed by SECOTP2D. This section contains the specific information required to run PRESECOTP2D including the input commands. It also contains specific information required to run SECOTP2D.

7.1 User interactions with PRESECOTP2D

PRESECOTP2D can be exercised interactively (Section 7.1.1) or from a command line (Section 7.1.2). Regardless of the approach, the user must specify up to eight files. A description of the eight files follows:

Files 1-3 and 6 are files that already exist. Files 4, 5, 7 and 8 are created by PRESECOTP2D. Files 4, 5 and 7 are used to run SECOTP2D. As files 1-3 and 6 already exist, they already have assigned names. Files 4-7 and 8 have names assigned by the user. A description of these files follows:

- File 1 is the input CAMDAT database that contains the grid and properties corresponding to the sampled vector needed for the transport simulation.

- File 2 is the input CAMDAT database that contains for the sampled vector the source information for each species being transported. This file is optional. This information may also be entered by hand in the PRESECOTP2D input file for simple functions.

- File 3 is the ASCII input file that controls PRESECOTP2D. It contains the commands that establish how SECOTP2D will be run and the information that will go into the input files that are created by PRESECOTP2D. Section 7.2.4 provides a detailed description of all the commands that make up this input file.

- File 4 is the ASCII input file created by PRESECOTP2D to run SECOTP2D for the sampled vector.

- File 5 is the binary file containing the property information used to run SECOTP2D for the sampled vector.

- File 6 is the optional input velocity file created by POSTSECOFL2D for the sampled vector to transfer double precision velocities from SECOFL2D to SECOTP2D without going through the CAMDAT database which is single precision. This is the preferred way to run SECOTP2D if the file is available.

- File 7 is the binary file containing velocities and source information used to run SECOTP2D for the sampled vector.

- File 8 is a diagnostic/debug file containing information about the PRESECOTP2D run. It is an optional file, but it is necessary that it be specified if the user wants to take advantage of the error reporting.

7.1.1 Exercising PRESECOTP2D Interactively

To execute PRESECOTP2D interactively, type PRESECOTP2D followed by a carriage return at the main menu of CAMCON or at the OpenVMS "$" prompt.

A banner scrolls down the screen and then the following information describing the file definitions is printed on the screen.

PRESECOTP2D expects the following files:

1) Input CAMDAT grid and material database
2) Input CAMDAT source database (optional)
3) PRESECOTP2D input file
4) SECOTP2D control input file
5) SECOTP2D property data input file
6) SECOFL2D transfer velocity data file (optional)
7) SECOTP2D velocity data input file
8) PRESECOTP2D diagnostic/debug file (optional)

The following prompts for filenames appear after the above list of files is printed on the screen. The file names in the angle brackets are the default names that will be chosen if only a carriage return is entered. Type "cancel" for no file to be chosen when a file name is optional.

The symbols `PRESECOTP2D$INPUT_DIRECTORY` and `PRESECOTP2D$OUTPUT_DIRECTORY` shown below may be assigned with the OpenVMS DEFINE command to be a specific directory. If they are not defined, the user must type [] to specify files in the current directory or [dir_spec], where [dir_spec] is the specification of a particular directory.

```
Enter the name of the Input CAMDAT database
<PRESECOTP2D$INPUT_DIRECTORY:CAMDAT.CDB> []flow_test.cdb

Enter the name of the Input CAMDAT source database (CANCEL for no file)
<PRESECOTP2D$INPUT_DIRECTORY:CAMDAT_S.CDB> []source_test.cdb

Enter the name of the PRESECOTP2D input file
<PRESECOTP2D$INPUT_DIRECTORY:PRESECOTP2D.INP> [Ipresecotp2d_test.inp

Enter the name of the SECOTP2D input file
<PRESECOTP2D$OUTPUT_DIRECTORY:SECOTP2D.INP> [Isecotp2d_test.inp

Enter the name of the SECOTP2D property data input file
<PRESECOTP2D$OUTPUT_DIRECTORY:PROPDAT.INP> []propdat_test.inp

Enter the name of the SECOFL2D transfer velocity data file (CANCEL for no
file)
<PRESECOTP2D$INPUT_DIRECTORY:VELOC.TRN> []veloc_test.trn

Enter the name of the SECOTP2D velocity data input file
```

```
<PRESECOTP2D$OUTPUT_DIRECTORY:VELDAT.INP> []veldat_test.inp

Enter the name of the PRESECOTP2D diagnostics/debug file (CANCEL for no file)
<PRESECOTp2D$OUTPUT_DIRECTORY:PRESECOTP2D.DBG> []presecotp2d_test.dbg
```

If the program completes without errors, the message

```
PRESECOTP2D Normal Completion
```

appears on the screen. If FORTRAN STOP appears on the screen, an error has occurred. The PRESECOTP2D diagnostics/debug file should be consulted to find a description of the error condition. This can be done using the editor.

7.1.2 Exercising PRESECOTP2D from a Command Line

To exercise PRESECOTP2D from a command line, type PRESECOTP2D at the OpenVMS prompt, but do not strike the carriage return key. Instead, follow the PRESECOTP2D command with the necessary filenames in the sequence indicated at the beginning of Section 6.0. (Up to 8 filenames are required [see above]; filenames not being used require the qualifier "cancel.") Use the hyphen ("-") at the end of lines on the computer screen as a continuation symbol. The operating system will read it to mean "continued without break on the next line." Thus, although the command procedure is typed on several lines, because of the hyphens, the computer reads it as being typed entirely on one line.

A sample command line procedure that executes PRESECOTP2D could be:

```
$ PRESECOTP2D flow_test.cdb source_test.cdb presecotp2d_test.inp
_$secotp2d_test.inp propdat_test.inp veloc_test.trn veldat_test.inp
_$presecotp2d_test.dbg
```

The command line may also be placed into a command file and the command file may be executed at the command line or in batch mode. In the command file, the continuation symbol may be used, but the "-$" would not appear at the beginning of the line.

7.2 Description Of Input Files

7.2.1 Input CAMDAT Database File

The data required to exist on the input CAMDAT database file is listed below.

Data Name	CAMDAT Default symbol	Data type
Aquifer properties		
Aquifer thickness	Thick	Attribute
Matrix solid properties (dual porosity)		
Porosity	Porosity	Attribute

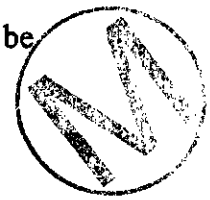
Tortuosity	Tortusty	Attribute
Retardation factors	Rtard_"symbol"	Property
Characteristic length of poros matrix blocks	Frctr_Sp	Property
Skin Resistance	SkinRest	Property
Fracture solid properties		
Porosity	Fporos	Attribute
Tortuosity	Ftortsty	Attribute
Dispersivity, longitudinal	Fdisplng	Property
Dispersivity, transverse	Fdisptrn	Property
Retardation factors	Frtrd_"symbol"	Property
Clay solid properties (dual porosity with optional clay lining)		
Porosity	Cporos	Attribute
Tortuosity	Ctortsty	Attribute
Retardation factors	Crtrd_"symbol"	Property
Clay thickness	FFilRato	Property
Initial conditions		
Concentration	Conc_"symbol"	Element
Velocity data		
Cell face centered Darcy x velocity	VelFacDX	Element
Cell face centered Darcy y velocity	VelFacDY	Element

7.2.2 SECOFL2D Transfer Velocity Data File

This is the input velocity file created by POSTSECOFL2D for the sampled vector. It is used to transfer double precision velocities from SECOFL2D to SECOTP2D without going through the CAMDAT database, which is single precision.

7.2.3 Input Source CAMDAT Database

This is an input database that contains the source function. The source function may also be entered manually in the input control file (Section 7.2.4).



7.2.4 Input Control File

Of the four input files to PRESECOTP2D described in Section 6.0, only one of them is created directly by the user. This file is the PRESECOTP2D ASCII input control file. This file provides input not only to PRESECOTP2D, it is the primary user interface to SECOTP2D also.

This section contains a description of the valid input that can be entered in the PRESECOTP2D ASCII input file. The format of the input file follows the standard format established for CAMCON input files as stated in the User's Reference Manual for CAMCON:

Compliance Assessment Methodology Controller, Version 3.0, section 2.9.4 [23]. The input file contains keywords, denoted by a leading asterisk (*), and parameters associated with the keywords that can function as secondary keywords or have input values associated with them. Most of the parameters are optional. If a parameter is not entered the default (in < >) will be taken. If "no default" is specified for a parameter, then a value must be entered if that parameter appears in the input file. The major keywords (denoted with a leading *) must appear on a line by itself. The second-level keywords and all of the associated parameters can appear on a single line. The upper case portion of the keywords and parameters shows the minimum character string required to be entered in the input file.

Twelve keywords are used in the PRESECOTP2D input file:

(1) *CONTROL_parameters,
(2) *VELOCITY_input,
(3) *OUTPUT,
(4) *TIME,
(5) *SPECIES_data,
(6) *PROPERTY_names,
(7) *BOUNDary_conditions,
(8) *SOURCE_term,
(9) *INITial_conditions,
(10) *DISCHARGE_BOUNDary,
(11) *DP _MESH
(12) *END

***CONTROL_parameters**

The *CONTROL keyword sets up some run control parameters. This keyword is optional. If the keyword is present, any parameter not specified in the input file will be assigned the default value.

MEDIUM, *type* <SINGLE>

type - indicates the type of medium; choices are:
SINGLE_porosity - for single-porosity medium
DUAL_porosity - for dual-porosity medium

TIME_SCHEME, *scheme* <EULER>

scheme- indicates the type of time scheme to use; choices are:
EULER - first order truncation error in time
TRAPezodial - second order truncation error in time
BACKward - (3-point) second order truncation error in time

SOURCE_COEFF, *variable_name*$_1$*=value*$_1$*, variable_name*$_2$*=value*$_2$*, ...*

- *variable_name*$_n$ is either

AX	<0.5>	X-coefficient of the source terms in the implicit part of the algorithm; the sum of AX and AY must be 1.
AY	<0.5>	Y-coefficient of the source terms in the implicit part of the algorithm; the sum of AX and AY must be 1.

LIMITER, *function* <MUSCL>

function - indicates the type of limiter function to use; choices are:
UPWIND - for an upwind scheme
CENTERED - for a centered scheme
MIXED=W0 - for a mixed upwind and centered scheme;
W0 sets the portion between the two;
W0=0 (upwind); W0=1 (centered)
MINMOD_1 - TVD limiter function; minmod(1,r); very dissipative
MINMOD_2 - TVD limiter function; minmod(1,2r)
MINMOD_3 - TVD limiter function; minmod(2,r)
MINMOD_4 - TVD limiter function; minmod(2,2r)
MUSCL - TVD limiter function; van Leer's MUSCL
SUBERBEE - TVD limiter function; Roe's superbee; highly compressive

CLIMATE, *name* <CLIMTIDX>

name - name of the CAMDAT variable containing the climate multiplier; must be a valid CAMDAT element variable name

***VELOCITY_input**

The *VELOCITY keyword controls how the velocities are read from the CAMDAT database. If the velocities are being read from the input velocity file generated by POSTSECOFL2D, this keyword is omitted. If the keyword is present, any parameter not specified in the input file will be assigned the default value.

X_DARCY, *name* <VELFACDX>

name - name of the CAMDAT variable containing the x-component of the face centered Darcy velocity; must be a valid CAMDAT element variable name

Y_DARCY, *name* <VELFACDY>

name - name of the CAMDAT variable containing the y-component of the face centered Darcy velocity; must be a valid CAMDAT element variable name

FLOW_CODE, *source* <SECO2>

source - identifies if the flow calculation was done the SECOFL2D code or another flow code to determine how the velocities for the ghost cells should be set up. If the flow calculation was done by the SECOFL2D code, these velocities get recorded on the database; otherwise, they are simply obtained from the nearest neighbor. Choices are:
SECO2 - for the SECOFL2D flow code
OTHER - for any other flow code calculation

STEP, *step* <1>

step - the step number to start reading the velocities from the CAMDAT database or the velocity transfer file. This number must be greater than 0 and must not exceed the number of steps on the database or the velocity transfer file.

STEADY, *steady* <NO>

steady - indicates if the flow field is steady state; choices are YES or NO; the STEP keyword can be used to select the desired time step to use the steady state velocity.

***OUTPUT**

The *OUTPUT keyword controls how the output is written to the binary output file and to the terminal. This keyword is optional. If the keyword is present, any parameter not specified in the input file will be assigned the default value.

STEP, *step* <1>

step - output printout control parameter for concentration; entering a value of n means that computed values of concentration are to be printed at every nth time step; must be greater than 0 and less than or equal to the number of time steps selected (see *TIME, NUMstep)

SCREEN_IO, *mode* <OFF>

mode - enables or disables iteration information to be written to the terminal during the SECOTP2D run; choices are: ON or OFF

***TIME**

The *TIME keyword sets up the time interval and the time steps for the SECOTP2D run. All of these parameters are considered by the code as optional, but in reality some of them are not. The user needs to choose appropriate values for setting up the time steps, start and stop time, etc. as desired.

NUM_step, *steps* <0>

steps - number of time steps for the transport calculation (does not have to match the number of steps on the velocity database as interpolation is done automatically); must be greater than 0.

TIME_GENeration, *mode* <AUTO>

mode - Specifies the method of time step generation; choices are:

AUTO - generates the time step using the equation (STOP_TIME - START_TIME)/(NUM_STEP - 1)

MANual - uses the specified DELTA_T for the time step; if DELTA_T*NUM_STEP is greater that STOP_TIME, the simulation ends at STOP_TIME; if DELTA_T*NUM_STEP is less than STOP_TIME, the simulation ends at DELTA_T*NUM_STEP

VAR - indicates that the time step is variable and will be calculated using the parameters entered with the VAR_DT keyword; delta_t(1)=INIT_DT, delta_t(n) = (delta_t((n-1)*GROW_FAC,MAX_DT)

DELTA_T, *time* <0.0>

time - the specified time step for manual time generation; must be greater than 0.

START_TIME, *time* <0.0>

time - initial time; must be equal to or greater than 0.

STOP_TIME, *time* <0.0>

time - end time; must be greater than 0.

VAR_DT, *variable_name*$_1$=*value*$_1$, *variable_name*$_2$=*value*$_2$, ...

- *variable_name*$_n$ is either

INIT_DT	<0.0>	- initial delta t; must be greater than 0.
GROW_FACtor	<0.0>	- delta t multiplier (growth factor); must be greater than 0.
MAX_DT	<0.0>	- maximum value for delta t; must be greater than 0.

***SPECIES_data**

The *SPECIES keyword sets up the radionuclides that will be transported. At least one radionuclide and at least one chain must be set up using these parameters. The default values are set to 0, but this value is never valid for any of the parameters. All of the parameters should appear in the input file for each radionuclide being set up.

NUClide, *variable_name*$_1$=*value*$_1$, *variable_name*$_2$=*value*$_2$, ...

- *variable_name*$_n$ is either

SYMBOL	<no default>	- name of the nuclide; usually the first one or two characters are the chemical symbol and the last three characters are the atomic mass unit of the isotope; only the leading alpha characters are used to match "symbol" for the retardation property names on the CAMDAT

database; any alpha numeric characters are valid; the length must not exceed 5.

INDEX	<0>	- component number of the nuclide; must be greater than 0.
LAMBDA	<0>	- decay constant of the nuclide ($\ln 2/t_{1/2}$); can be either a value or a CAMDAT database variable name containing the decay constant; if a value is entered it must match the number on the Secondary Database
FREE_H2O_DIFF	<0>	- free water molecular diffusion; used with the tortuosity to calculate the molecular diffusion; can be either a value or a CAMDAT database variable name containing the decay constant; if a value is entered it must match the number on the Secondary Database
CURIE	<0>	- conversion factor for converting mass flux to activity (Cu/kg); can be either a value or a CAMDAT database variable name containing the decay constant; if a value is entered it must match the number on the Secondary Database

CHAIN_definition, *variable_name*$_1$=*value*$_1$, *variable_name*$_2$=*value*$_2$, ...

- *variable_name*$_n$ is either

CHAIN_NUMber	<0>	- chain number; must be greater than 0; chain number must be unique.
NUM_SPECIES	<0>	- number of species in this chain; must be greater than 0.
NUC_INDICES	<0>	- list of indices (from INDEX above) of the species in this chain in proper order; must all be greater than 0.

***PROPerty_names**

The *PROP keyword establishes what CAMDAT variable names contains the material properties of the transport medium. All of these parameters are optional. If any of them are not specified in the input file, the default value will be taken. If the default value is used, that name must be a valid name on the CAMDAT database.

AQUIFER, *variable_name$_1$=value$_1$, variable_name$_2$=value$_2$, ...*

- *variable_name$_n$* is either

THICKness	<THICK>	- CAMDAT attribute name for aquifer thickness; must be a valid CAMDAT variable name

DIFFusive, *variable_name$_1$=value$_1$, variable_name$_2$=value$_2$, ...*

- *variable_name$_n$* is either

TORT	<TORT>	- CAMDAT attribute name for the diffusive tortuosity of the matrix material; must be a valid CAMDAT variable name
POROSITY	<POROSITY>	- CAMDAT attribute name for the diffusive porosity of the matrix material; must be a valid CAMDAT variable name
RETARD_factor	<RTARD_"symbol">	- CAMDAT property name for the diffusive retardation factor of the nuclide with the chemical symbol "symbol" in the porous matrix material; the default name is automatically generated using the nuclides listed on the NUClide cards under the *SPECIES keyword; must be a valid CAMDAT variable name

DUAL_porosity, *variable_name$_1$=value$_1$, variable_name$_2$=value$_2$, ...*

- *variable_name$_n$* is either

BLOCK_LENgth	<FRCTR_SP>	- CAMDAT property name for the half length of the porous matrix blocks for a dual porosity medium; must be a valid CAMDAT variable name
SKIN_RESIStance	<SKINREST>	- CAMDAT property name for the skin resistance; must be a valid CAMDAT variable name

CLAY_THICKness	<FFILRATO>	- CAMDAT attribute name for the ratio of the clay thickness to the fracture aperture; must be a valid CAMDAT variable name

ADVECtive, *variable_name$_1$=value$_1$, variable_name$_2$=value$_2$, ...*

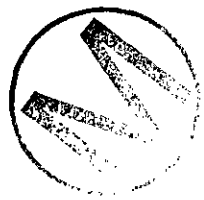
- *variable_name$_n$* is either

DISP_LNG	<FDISPLNG>	- CAMDAT attribute name for longitudinal dispersivity; must be a valid CAMDAT variable name
DISP_TRN	<FDISPTRN>	- CAMDAT attribute name transverse dispersivity; must be a valid CAMDAT variable name
TORT	<FTORT>	- CAMDAT attribute name for the diffusive tortuosity of the fractured material; must be a valid CAMDAT variable name
POROSITY	<FPOROS>	- CAMDAT attribute name for the diffusive porosity of the fractured material; must be a valid CAMDAT variable name
RETARD_factor	<FRTRD_"symbol">	- CAMDAT property name for the diffusive retardation factor of the nuclide with the chemical symbol "symbol" in the fractured material; the default name is automatically generated using the nuclides listed on the NUClide cards under the *SPECIES keyword; must be a valid CAMDAT variable name

CLAY_LINING, *variable_name$_1$=value$_1$, variable_name$_2$=value$_2$, ...*

- *variable_name$_n$* is either

TORT	<TORTCLAY>	- CAMDAT attribute name for tortuosity of the clay lining; used only for dual porosity models; must be a valid CAMDAT variable name



POROSITY	<PORCLAY>	- CAMDAT attribute name for the porosity of the clay lining; must be a valid CAMDAT variable name
RETARD_factor	<FRTDC_"symbol">	- CAMDAT property name for the retardation factor of the nuclide with the chemical symbol "symbol" in the clay lining; the default name is automatically generated using the nuclides listed on the NUClide cards under the *SPECIES keyword; must be a valid CAMDAT variable name

***BOUNDary_conditions**

The *BOUND keyword sets up the boundary conditions for the SECOTP2D run. This keyword is optional. If it is not included, the default of "AUTO" is set for all species and all boundaries. If the keyword is present, any parameter not specified in the input file will be assigned the default value. Any parameter that has no default designated is not optional and must be included.

TOP, *variable_name*$_1$=*value*$_1$, *variable_name*$_2$=*value*$_2$, ...

- *variable_name*$_n$ is either

TYPE	<AUTO>	- type of boundary condition; choices are: AUTO - SECOTP2D automatically selects the type based on the velocity field NEUMAN - for a Neumann boundary condition DIRichlet - for a Dirichlet boundary condition
VALUE	<0.0>	- value of concentration or gradient; must be equal or greater than 0
SYMBOL	<no default>	- name of the nuclide associated with the specified concentration; must be a valid symbol name
NRANGE	<0,0>	- range of elements for which this boundary condition applies; must be greater than 0 and less than or equal to number of elements in the x-direction
INCR	<1>	- increment for element range in NRANGE; must be greater than 0 and less than or equal to the number of elements in the x-direction

BOTtom, *variable_name$_1$=value$_1$, variable_name$_2$=value$_2$, ...*

- *variable_name$_n$* is either

TYPE	<AUTO>	- type of boundary condition; choices are: AUTO - SECOTP2D automatically selects the type based on the velocity field NEUMAN - for a Neumann boundary condition DIRichlet - for a Dirichlet boundary condition
VALUE	<0.0>	- value of concentration or gradient; must be equal or greater than 0
SYMBOL	<no default>	- name of the nuclide associated with the specified concentration; must be a valid symbol name
NRANGE	<0,0>	- range of elements for which this boundary condition applies; must be greater than 0 and less than or equal to number of elements in the x-direction
INCR	<1>	- increment for element range in NRANGE; must be greater than 0 and less than or equal to the number of elements in the x-direction

LEFT, *variable_name$_1$=value$_1$, variable_name$_2$=value$_2$, ...*

- *variable_name$_n$* is either

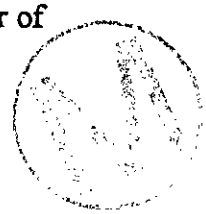
TYPE	<AUTO>	- type of boundary condition; choices are: AUTO - SECOTP2D automatically selects the type based on the velocity field NEUMAN - for a Neumann boundary condition DIRichlet - for a Dirichlet boundary condition
VALUE	<0.0>	- value of concentration or gradient; must be equal or greater than 0
SYMBOL	<no default>	- name of the nuclide associated with the specified concentration; must be a valid symbol name
NRANGE	<0,0>	- range of elements for which this boundary condition applies; must be greater than 0 and less than or equal to number of elements in the x-direction

INCR	<1>	- increment for element range in NRANGE; must be greater than 0 and less than or equal to the number of elements in the x-direction

RIGHT, *variable_name$_1$=value$_1$, variable_name$_2$=value$_2$, ...*

- *variable_name$_n$* is either

TYPE	<AUTO>	- type of boundary condition; choices are: AUTO - SECOTP2D automatically selects the type based on the velocity field NEUMAN - for a Neumann boundary condition DIRichlet - for a Dirichlet boundary condition
VALUE	<0.0>	- value of concentration or gradient; must be equal or greater than 0
SYMBOL	<no default>	- name of the nuclide associated with the specified concentration; must be a valid symbol name
NRANGE	<0,0>	- range of elements for which this boundary condition applies; must be greater than 0 and less than or equal to number of elements in the x-direction
INCR	<1>	- increment for element range in NRANGE; must be greater than 0 and less than or equal to the number of elements in the x-direction



***SOURCE_term**

The *SOURCE keyword sets up the source definition for each radionuclide. A source function must be set up for each radionuclide either by obtaining it from the source database or by manually setting up the times and function values. The function may be identically zero.

SOURCE, *source*

source - set to CAMDAT if the source term times and function values are to be read from the history variables of the source CAMDAT database; there are no other choices.

TERM_DEFinition, *variable_name$_1$=value$_1$, variable_name$_2$=value$_2$, ...*

- *variable_name$_n$* is either

SYMBOL	<no default>	- name of the nuclide for which the source term is specified; must be a valid symbol name
NAME_SOL	<no default>	- name of CAMDAT history variable that contains the integrated source term information; must be a valid CAMDAT variable name
NAME_FLUX	<no default>	- name of CAMDAT history variable that contains the source term flux information; must be a valid CAMDAT variable name
NUM_POINTS	<0>	- number of control points on the graph of the function versus time; or the number of time steps in the source CAMDAT database; must be greater than 0
TIMES	<0.0>	- array of time values corresponding to control points on the graph of function versus time; first time must be greater than simulation start time; not required if the source data is read from a CAMDAT database
VALUES	<0.0>	- array of integrated function values corresponding to the TIMES; all values must be greater than or equal to 0; not required if the source data is read from a CAMDAT database
IRANGE	<0,0>	- range of elements in the i-direction to apply the source term described in this definition; must be greater than 0 and less than or equal to the number of elements in the x direction.
JRANGE	<0,0>	- range of elements in the j-direction to apply the source term described in this definition; must be greater than 0 and less than or equal to the number of elements in the y direction.

***INITial_conditions**

The *INIT keyword sets the initial conditions for the radionuclides. This keyword is optional. If it is not present, the initial concentration of each radionuclide is set to 0. If the keyword is present, any parameter not specified will in the input file be assigned the default value. Any parameter that has no default designated is not optional and must be included.

TIME_STEP, *time* <0.0>

time - time step from CAMDAT to read the initial conditions; must be greater than 0 and less than or equal to the number of steps on the input CAMDAT database

DEFinition, *variable_name*$_1$=*value*$_1$, *variable_name*$_2$=*value*$_2$, ...

- *variable_name*$_n$ is either

NAME	<CONC_"symbol">	- CAMDAT element variable name storing initial value of concentration for the nuclide specified by "symbol"; the default name will be automatically generated using the nuclides listed on the NUClide cards under the *SPECIES keyword; must be a valid CAMDAT variable name.
INCrement	<1>	- element increment for elements specified by NRANGE; must be greater than 0 and less than or equal to the number of elements
NRANGE	<0,0>	- range of elements where the value CONC applies; must be greater than 0 and less than the number of elements
SYMBOL	<no default>	- name of the nuclide for which the initial value of CONC or DEF is specified; must be a valid symbol name
VALUE	<0>	- concentration value used to overwrite the CAMDAT value or used for nodes defined by NRANGE; must be greater than or equal to 0

***DISCHARGE_BOUNDary**

The *DISCHARGE_BOUND keyword sets up rectangular discharge boundaries around which the source is integrated. This keyword is optional. If it is specified in the input file, appropriate values must be entered for each of the parameters.

NUM_BNDS, *number* <0>

number - number of discharge boundaries to set up; must be greater than or equal to 0

BOUND_DEFinition, *variable_name*$_1$*=value*$_1$*, variable_name*$_2$*=value*$_2$*, ...*

- *variable_name*$_n$ is either

TOP_LEFT	<0,0>	- i, j coordinate of top left corner of the rectangular discharge boundary; must be contained in the grid
BOTTOM_RIGHT	<0,0>	- i, j coordinate of bottom right corner of the rectangular discharge boundary; must be contained in the grid

***DP_MESH**

The *DP_MESH keyword is used to enter data describing the discretization of the porous matrix blocks used for dual porosity. This keyword must be present for dual porosity runs. It will not be present for single porosity runs.

AUTO, *variable_name*$_1$*=value*$_1$*, variable_name*$_2$*=value*$_2$*, ...*

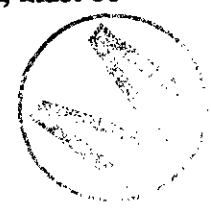
- *variable_name*$_n$ is either

INIT_DIST	<0.0>	- initial non-dimensional cell size at the matrix fracture interface; must be greater than 0 and less than 1
NUM_NODES	<0>	- number of one dimensional nodal points in each grid line representing the porous matrix domain; must be greater than 0
CLAY_FRAC	<0.0>	- initial cell size for fractures with a clay lining; units are a fraction of the clay lining (i.e., .3, .15, etc.); must be greater than 0 and less than 1

MANual, *variable_name*$_1$*=value*$_1$*, variable_name*$_2$*=value*$_2$*, ...*

- *variable_name*$_n$ is either

NUM_NODES	<0>	- number of one dimensional nodal points in each grid line representing the porous matrix domain; must be greater than 0



DISTances	<0>	- list of nodal values of dimensionless distance from the center of the matrix block; this dimensionless distance is defined as the ratio of the actual distance coordinate to the half-thickness of the block; all must be less than 1 and monotonically increasing.

***END**

The *END keyword signals the end of the input file. There are no other keywords or parameters associated with it.

SUMMARY

The following summarizes all PRESECOTP2D input keyword and parameters:

---

*CONTROL_parameters

1. MEDIUM=
2. TIME_SCHEME=
3. SOURCE_COEFF, AX=, AY=
4. LIMITER=
5. CLIMATE=

---

*VELOCITY_input

1. X_DARCY=
2. Y_DARCY=
3. FLOW_CODE=
4. STEP=

---

*OUTPUT

1. STEP=

2. SCREEN_IO=

---

*TIME

1. NUM_step=
2. TIME_GENeration=
3. DELTA_T=
4. START_TIME=
5. STOP_TIME=
6. VAR_DT, INIT_DT=, GROW_FACtor=, MAX_DT=

---

*SPECIES_data

1. NUClide, SYMBOL=, INDEX=, LAMBDA=, FREE_H2O_DIFF=, CURIE=
2. CHAIN_definition, CHAIN_NUMber=, NUM_SPECIES=, NUC_INDICES=

---

*PROPerty_names

1. AQUIFER, THICKness=
2. DIFFusive, TORT=, POROSITY=, RETARD_factor=
2. DUAL_porosity, BLOCK_LENgth=, SKIN_RESIStance=, CLAY_THICKness=
3. ADVECtive, DISP_LNG=, DISP_TRN=, TORT=, POROSITY=, RETARD_factor=
4. CLAY_LINING, TORT=, POROSITY=, RETARD_factor=

---

*BOUNDary_conditions

1. TOP, TYPE=, VALUE=, SYMBOL=, NRANGE=, INCR=
2. BOTtom, TYPE=, VALUE=, SYMBOL=, NRANGE=, INCR=

3. RIGHT, TYPE=, VALUE=, SYMBOL=, NRANGE=, INCR=

4. LEFT, TYPE=, VALUE=, SYMBOL=, NRANGE=, INCR=

---

*SOURCE_term

1. SOURCE=

2. TERM_DEFinition, SYMBOL=, NAME_SOL=, NAME_FLUX=, NUM_POINTS=, TIMES=, VALUES=, IRANGE=, JRANGE=

---

*INITial_conditions

1. TIME_STEP=

2. DEFinition, NAME=, INCrement=, NRANGE=, SYMBOL=, VALUE=

---

*DISCHARGE_BOUNDary

1. NUM_BNDS=

2. BOUND_DEFinition, TOP_LEFT=, BOTTOM_RIGHT=

---

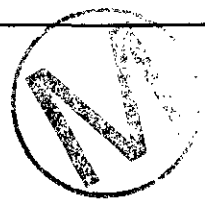
*DP_MESH

1. AUTO, INIT_DIST=, NUM_NODES=, CLAY_FRAC=

2. MANual, NUM_NODES=, DISTances=

---

*END



7.3 Input File for PRESECOTP2D

The following is an example input file for PRESECOTP2D.

```
!=============================================================
!          CHAIN 1 PU240
!          CHAIN 2 PU239
!          CHAIN 3 AM241 -> NP237 -> U233
```

```
!              CHAIN 4  U234 -> TH230
!=============================================================================!
*CONTROL
  MEDIUM=SINGLE
  TIME_SCHEME=TRAP
!
*OUTPUT
  STEP=100
!
*TIME
  NUM_STEPS=1000
  START_TIME=3.1558E+10
  STOP_TIME=3.1558E+11
!
*SPECIES
  NUCLIDE, SYMBOL=PU240, INDEX=1, LAMBDA=3.36E-12,
           FREE_H2O_DIFF=0.48E-10
  NUCLIDE, SYMBOL=PU239, INDEX=2, LAMBDA=9.13E-13,
           FREE_H2O_DIFF=0.48E-10
  NUCLIDE, SYMBOL=AM241, INDEX=3, LAMBDA=5.08E-11,
           FREE_H2O_DIFF=1.76E-10
  NUCLIDE, SYMBOL=NP237, INDEX=4, LAMBDA=1.03E-14,
           FREE_H2O_DIFF=1.76E-10
  NUCLIDE, SYMBOL=U233,  INDEX=5, LAMBDA=1.39E-13,
           FREE_H2O_DIFF=1.70E-10
  NUCLIDE, SYMBOL=U234,  INDEX=6, LAMBDA=8.98E-14,
           FREE_H2O_DIFF=2.70E-10
  NUCLIDE, SYMBOL=TH230, INDEX=7, LAMBDA=2.85E-13,
           FREE_H2O_DIFF=1.00E-10
!          CHAIN 1 PU240
!          CHAIN 2 PU239
!          CHAIN 3 AM241 ->NP237 -> U233
!          CHAIN 4  U234 ->TH230
  CHAIN CHAIN_NUM=1, NUM_SPECIES=1, NUC_INDICES=1
  CHAIN CHAIN_NUM=2, NUM_SPECIES=1, NUC_INDICES=2
  CHAIN CHAIN_NUM=3, NUM_SPECIES=3, NUC_INDICES=3,4,5
  CHAIN CHAIN_NUM=4, NUM_SPECIES=2, NUC_INDICES=6,7
!
*PROPTERTY_NAMES
 ADVECTIVE   TORT=FTORT, POROSITY=FPOROS, RETARD=FRTRD
!
*SOURCE_TERM
SOURCE = CAMDAT
TERM_DEF NUM_POINTS=192, NAME_SOL= M00PU240, SYMBOL=PU240, &
  IRANGE=13,13 JRANGE=43,43
```

```
 TERM_DEF NUM_POINTS=192, NAME_SOL= M00PU239, SYMBOL=PU239, &
   IRANGE=13,13 JRANGE=43,43
 TERM_DEF NUM_POINTS=192, NAME_SOL= M00AM241, SYMBOL=AM241, &
   IRANGE=13,13 JRANGE=43,43
 TERM_DEF NUM_POINTS=192, NAME_SOL= M00NP237, SYMBOL=NP237, &
   IRANGE=13,13 JRANGE=43,43
 TERM_DEF NUM_POINTS=192, NAME_SOL= M00U233,  SYMBOL=U233, &
   IRANGE=13,13 JRANGE=43,43
 TERM_DEF NUM_POINTS=192, NAME_SOL= M00U234,  SYMBOL=U234, &
   IRANGE=13,13 JRANGE=43,43
 TERM_DEF NUM_POINTS=192, NAME_SOL= M00TH230, SYMBOL=TH230, &
   IRANGE=13,13 JRANGE=43,43
*END
!=============================================================
```

7.4 INPUT PARAMETER CHECKING

All of the keywords and parameters entered in the PRESECOTP2D ASCII input file are checked by PRESECOTP2D for validity. The input values associated with the parameters are *not* all checked for validity. The values that *are* checked by the code will cause the execution to abort if an invalid value is entered. An error message is written to the diagnostic/debug output file describing the parameter and the value that generated the error.

The following is a list of the input values that are *not* checked by the code for a valid range.

```
*CONTROL_PARAMETERS
     SOURCE_COEFF, AX =
                   AY =
*OUTPUT
     STEP =
*TIME
     NUM_STEP =
     DELTA_T =
     START_TIME =
     STOP_TIME =
     VAR_DT, INIT_DT =
             GROW_FACTOR =
             MAX_DT =
*SPECIES_DATA
     CHAIN_DEFINITION, CHAIN_NUMBER =
                       NUM_SPECIES =
                       NUC_INDICES =
*BOUNDARY_CONDITIONS
     TOP, VALUE =
          NRANGE =
```

```
        INCR =
    BOT, VALUE =
        NRANGE =
        INCR =
    RIGHT, VALUE
        NRANGE =
        INCR =
    LEFT, VALUE =
        NRANGE =
        INCR =
*SOURCE_TERM
    TERM_DEFINITION, NUM_POINTS =
                     TIMES =
                     VALUES =
                     IRANGE =
                     JRANGE =
*INITIAL_CONDITIONS
    TIME_STEP =
    DEFINITION, INC =
                NRANGE =
                VALUE =
*DISCHARGE_BOUNDARIES
    NUM_BNDS =
    BOUND_DEFINITION, TOP_LEFT =
                      BOTTOM_RIGHT =
*DP_MESH
    AUTO, INIT_DIST =
        NUM_NODES =
        CLAY_FRAC =
    MAN, NUM_NODES =
        DISTANCES =
```

7.5 User Interactions With SECOTP2D

To execute SECOTP2D for a sampled vector, type SECOTP2D at the Alpha system "$" prompt. SECOTP2D will request the names of three files. Alternatively, the user may append the names of the three files (in the order listed below) to the SECOTP2D command line. The three files that the user must specify are listed below:

1. The input control file. This file specifies the processing options for SECOTP2D. Unlike the input control files for most of the WIPP PA codes, the user has no interactive control over the contents of this file. This file is generated by the upstream code PRESECOTP2D. All of the interaction with this file is done through choices made in the input file to the preprocessor. This file in turn contains the names of the binary property and velocity files needed to run this code (see below).

2. The binary output data file. This file contains the output of SECOTP2D in binary format. This file is processed by POSTSECOTP2D.

3. The diagnostics/debug file. This file contains run-time information on the execution of SECOTP2D.

In addition, two other input files are required for SECOTP2D to run. These files, listed below, are specified by the user when the preprocessor is run.

- *Property data input file. This file contains the property information output from* PRESECOTP2D in binary format, which is used to run SECOTP2D.

- Velocity data input file. This file contains the velocities and source information output from PRESECOTP2D in binary fon-nat used to run SECOTP2D.

7.6 Description Of Input Files

7.6.1 Input Control File

The input control file to SECOTP2D is generated by PRESECOTP2D based in part on its own input control file. For all practical purposes, the code-generated input control file to SECOTP2D is transparent to the user.

7.6.2 Property Data Input and Velocity Data Input Files

The property data input file, which contains CAMDAT property information, is binary and is not user-specified. It is therefore transparent to the user.
The velocity data input file, which contains CAMDAT velocities and source information, is binary and is not user-specified. It is therefore transparent to the user.

7.7 Description Of Output Files

7.7.1 Binary Output File

The binary output file contains the output of SECOTP2D in binary format. Because it is binary, it cannot be read by the user.

7.7.2 Diagnostics/Debug Output File

A sample diagnostics/debug file, the only output file from SECOTP2D that can be read by the user, is provided in Appendix II.

7.8 Postprocessor, POSTSECOTP2D

POSTSECOTP2D is a postprocessor that writes SECOTP2D output to a CAMDAT database. It writes up to three types of data to the database. These are 1) face centered Darcy velocities (specific discharges) that were used by the simulation, 2) concentration at each element in the grid for each radionuclide transported, and 3) discharge information for each radionuclide transported and each discharge boundary set up. Discharge boundaries are optional and this information is written if one or more discharge boundaries are set up.
The following is a list of the input and output files that are needed to run POSTSECOTP2D.

1. Input CAMDAT database
2. SECOTP2D binary output file
3. Output CAMDAT database
4. POSTSECOTP2D diagnostic/debug file (optional)

The first file is the input CAMDAT database that was used by PRESECOTP2D. It contains property and grid information. The second file is the binary output file from SECOTP2D. The third file is the output database created by POSTSECOTP2D. It will contain the information from the input database and also the results from SECOTP2D. The fourth file is an optional diagnostic/debug file created by POSTSECOTP2D that contains information and error messages generated by running POSTSECOTP2D. POSTSECOTP2D does not require an ASCII input command file as did PRESECOTP2D.

8 TEST PROBLEM

To run SECOTP2D in the CAMCON environment, first a preprocessor, PRESECOTP2D, must be run. Then at the end of the calculation a postprocessor, POSTSECOTP2D, must be run. The preprocessor creates the necessary input files for SECOTP2D by translating data from an input CAMDAT database to binary property and velocity files. The postprocessor creates an output CAMDAT database by adding the results of the run to the input CAMDAT database. The following test problem gives an example of how to run each of the separate three codes.

This test problem is one of the 1992 WIPP PA vectors [17]. It is discretized by a 46x53 grid. The input velocity field was generated by SECOFL2D.

8.1 PRESECOTP2D

The purpose of running PRESECOTP2D is to produce the necessary input files so that SECOTP2D may be run. To run PRESECOTP2D, CAMDAT databases are needed as well as an ASCII input file.

A copy of the PRESECOTP2D ASCII input file, PRESECOTP2D TEST.INP follows. VELOC_TEST.TRN is an input velocity file that was created by POSTSECOFL2D. The binary files created by PRESECOTP2D, PROPDAT_TEST.INP and VELDAT_TEST.INP cannot be printed, but can be found in the directory CAMCON$ROOT: [PRESECOTP2D .TEST].

The test case for PRESECOTP2D can be run on the Alpha (Beatle) in the directory user's the following file assignments:

Input CAMDAT database:
CAMCON$ROOT:[PRESECOTP2D.TEST]FLOW_TEST.CDB

Input CAMDAT source database:
CAMCON$ROOT:[PRESECOTP2D.TEST]SOURCE_TEST.CDB

PRESECOTP2D input filename:
CAMCON$ROOT:[PRESECOTP2D.TEST]PRESECOTP2D_TEST.INP

SECOTP2D control input filename:
SECOTP2D_TEST.INP

SECOTP2D property data input filename:
PROPDAT_TEST.INP

SECOFL2D transfer velocity data filename:
CAMCON$ROOT:[PRESECOTP2D.TEST]VELOC_TEST.TRN

SECOTP2D velocity data input filename:
VELDAT_TEST.INP

PRESECOTP2D diagnostics/debug filename:
PRESECOTP2D_TEST.DBG

```
!=============================================================
! Test case for SECOTP2D
!=============================================================
*CONTROL
   MEDIUM=dual
   TIME_SCHEME=back
   LIMITER=muscl
*VELOCITY
   STEP=1
*OUTPUT
   STEP=50
   SCREEN_IO=ON
*TIME
   NUM_STEP=500
   TIME_GEN=AUTO
   START_TIME=3.15569E10
   STOP_TIME =3.15569E11
*SPECIES
      NUCLIDE SYMBOL=U233, INDEX=1, LAMBDA=1.39E-13, &
         FREE_H20_DIFF=1.70E-10, CURIE=9.68
      CHAIN CHAIN_NUM=1 NUM_SPECIES=1 NUC_INDICES=1
*PROPERTY
      DIFFUS, TORT= MTORT, POROSITY=MPOROS, RETARD=ZRTRD
      DUAL BLOCK_LEN=BLOCKLEN SKIN_RESIST= PZERO
      ADVECTIVE DISP_LNG=DISP_LNG, DISP_TRN=DISP_TRN, TORT= FTORT, &
            POROSITY=FPOROS, RETARD=ZRTRD
*SOURCE
      SOURCE = CAMDAT
      TERM_DEF SYMBOL=U233, NAME_SOL=M00U233, NUM POINTS= 186, &
            IRANGE= 21,21, JRANGE=45,45
*DISCHARGE_BOUND
   NUM_BNDS=3
   BOUND_DEF TOP_LEFT=1,53, BOTTOM_RIGHT=46,3
   BOUND_DEF TOP_LEFT=1,53, BOTTOM_RIGHT=46,23
   BOUND_DEF TOP_LEFT=1,53, BOTTOM_RIGHT=46,34
*DP_MESH
      AUTO INIT_DIST=.01, NUM_NODES=10
*END
```

8.2 SECOTP2D

This test case for SECOTP2D was chosen to exercise features of the code that are used while running SECOTP2D for the PA calculation. This example exercises the dual porosity capability of the code. There is one chain with one species. The binary files, PROPDAT_TEST.INP and VELDAT_TEST.INP, are input files. The binary files SECOTP2D_TEST.BIN, and the ASCII file, SECOTP2D_TEST.OUT, are output files for SECOTP2D. The input files can be found in the directory CAMCON$ROOT:[SECOTP2D.TEST].

The test case for SECOTP2D can be run on the Alpha (Beatle) in the directory using the following file assignments:

SECOTP2D input filename:
CAMCON$ROOT:[SECOTP2D.TEST]SECOTP2D_TEST.INP

SECOTP2D binary output filename:
SECOTP2D_TEST.BIN

ENTER SECOTP2D output filename:
SECOTP2D_TEST.OUT

8.3 POSTSECOTP2D

The purpose of running POSTSECOTP2D is to add the results from a SECOTP2D run to a CAMDAT database. The output results file is binary as are the CAMDAT databases. The data on the CAMDAT database may be examined or plotted using BLOT [22]. The test case for POSTSECOTP2D can be run in the user's directory using the following file assignments:

Input CAMDAT database:
CAMCON$ROOT:[POSTSECOTP2D.TEST]FLOW TEST.CDB

SECOTP2D binary output filename:
SECOTP2D_TEST.BIN

Output CAMDAT database:
SECOTP2D_TEST.CDB

POSTSECOTP2D diagnostics/debug filename:
POSTSECOTP2D_TEST.DBG

A plot for this example showing the contours of concentration for CONU233 at TIME = 10,000 years, created using BLOT, is shown in Figure 26.

REFERENCES

[1] Roache, P. J., 'Computational Fluid Dynamics Algorithms Developed for WIPP Site Simulations,' Proceedings IX International Conference on Computational Methods in Water Resources, Denver, Colorado, 9-12 June 1992, T. Russell, et al., eds., pp. 375-382.

[2] Roache, P. J., 'Computational Fluid Dynamics Algorithms and Codes Developed for WIPP Site Simulations,' Proceedings Asian Pacific Conference on Computational Mechanics, 11-13 December 1991, University of Hong Kong, J.H.W. Lee, et al., eds., H. Balkeema, Amsterdam.

[3] Salari, K., Knupp, P., Roache, P., and Steinberg, S., 'TVD Applied to Radionuclide Transport in Fractured Porous Media,' Proceedings IX International Conference on Computational Methods in Water Resources, Denver, Colorado, 9-12 June 1992, T. Russell, et al., eds., pp. 141-148.

[4] Roache, P. J., 'The SECO Suite of Codes for Site Performance Assessment,' Proceedings 1993 Intl. High-Level Radioactive Waste Management Conf., April 26-30, 1993, Las Vegas, NV.

[5] Huyakorn, P. S. and Pinder, G. F., *Computational Methods in Subsurface Flow*, Academic Press, New York, 1983.

[6] Huyakorn, P. S., Lester, B. H., and Mercer, J. W. 'An Efficient Finite Element Technique for Modeling Transport in Fractured Porous Media: Single Species Transport,' *Water Res. Res.*, Vol. 19, No. 3, pp. 841-854, 1983.

[7] Bear, J. and Bachmat, Y. *Introduction to Modeling of Transport Phenomena in Porous Media*, Kluwer Academic Publishers, Dordrecht, Netherlands, 1990.

[8] Scheidegger, A. E., *The Physics of Flow Through Porous Media*, University of Toronto Press, 1974.

[9] Streltsova-Adams, T. D., 'Well Hydraulics in Heterogeneous Aquifer Formations,' *Advances in Hydroscience*, Vol. 11, (Ed., Chow, V.T.), pp. 357-423 Academic Press, New York, 1978.

[10]Fletcher, C. A. J., *Computational Techniques for Fluid Dynamics*, Vol. I and II, Springer-Verlag, 1988.

[11]Sweby, P.K., 'High Resolution Schemes Using Flux Limiters for Hyperbolic Conservation Laws', *SIAM J. Numer. Anal.* 21: 995-1011.

[12]Yee, H. C., 'Construction of Explicit and Implicit Symmetric TVD Schemes and Their Applications,' *J. Comp. Phys.*, Vol. 68, pp. 151-179, 1987.

[13]Tang, D. H., Frind, E. O., and Sudicky, E. A. 'Contaminant Transport in Fractured Porous Media: Analytical Solution for a Single Fracture,' *Water Resources Research*, Vol. 17, No. 3, pp. 555-564, 1981.

[14]Lester, D. H., Jansen G., and Burkholder, H. C., 'Migration of Radionuclide Chains Through an Adsorbing Medium,' AIChE Symposium series, Vol. 71, No. 152.

[15]Roache, P. J., 'A Method for Uniform Reporting of Grid Refinement Studies,' ASME FED-Vol. 158, Quantification of Uncertainty in Computational Fluid Dynamics, ASME Fluids Engineering Division Summer Meeting, Washington, DC, 20-24 June 1993. Celik., I, Chen, C. J. Roache, P. J., and Scheurer, G. Eds., pp. 109-120.

[16]Roache, P. J., 'A Method for Uniform Reporting of Grid Refinement Studies,' Proc. 11th AIAA Computational Fluid Dynamics Conference, July 6-9, 1993, Orlando, Fl., Part 2, pp. 1-57-1058.

[17]Roache, P. J., 'Perspective: A Method for Uniform Reporting of Grid Refinement Studies,' ASME Journal of Fluids Engineering, Vol. 116, No. 3, Sept.1994, pp. 405-413.

[18]Anon., 'Preliminary Performance Assessment for the Waste Isolation Pilot Plant,' Sandia Report SAND92-0700/4, December 1992.

[19]Sudicky, E. A. and Frind, E. O., 'Contaminant Transport in Fractured Porous Media: Analytical Solutions for a System of Parallel Fractures,' *Water Resources Research*, Vol. 18, No. 6, pp. 1634-1642, 1982.

[20]Davis, G. B. and Johnston, C. D., Comment on "Contaminant Transport in Fractured Porous Media: Analytical Solutions for a System of Parallel Fractures" by Sudicky, E. A., and Frind, E. O., *Water Resources Research*, Vol. 20, No. 9, pp. 1321-1322, Sept. 1984.

[21]van Gulick, P., 'Evaluation of Analytical Solution for a System of Parallel Fractures for Contaminant Transport in Fractured Porous Media,' Sandia Report SAND 942877, to appear. Submitted for publication to Water Resources Research.

[22]Javandel, I., Doughty, C., and Tsang, C. F. *Groundwater Transport: Handbook of Mathematical Models*, American Geophysical Union, Washington, D.C., 1984.

[23]Rechard, R. P., ed., 'User's Reference Manual for CAMCON: Compliance Assessment Methodology Controller Version 3.0,' Sandia Report SAND90-1983 UC-721, August 1992.

APPENDIX
2D Transport Benchmark Solutions - I

P. Knupp
Ecodynamics Research Associates, P.O. Box 9229,
Albuquerque, NM 87119

The following tests the basic 2D transient algorithm for fracture transport with advection, diffusion, and decay.

The governing equation for this benchmark is:

$$\begin{aligned}&\left(D_{11}c_x + D_{12}c_y - uc\right)_x + \\ &\left(D_{12}c_x + D_{22}c_y - vc\right)_y = \phi R c_t + \phi R \lambda c + g(x,y,t)\end{aligned} \tag{1}$$

where

$$|\mathbf{v}|D_{ij} = \alpha_T |\mathbf{v}|^2 \delta_{ij} + (\alpha_L - \alpha_T) v_i v_j. \tag{2}$$

This equation is obtained from the more general equation solved in the transport code by assuming

- Free-water molecular diffusion is zero,
- Dual porosity term is off,
- Single-species decay.

One way to construct a solution which has time-independent boundary conditions is to assume the concentration is of the form

$$c(x,y,t) = c_M p(x,y) e^{-\lambda t}, \tag{3}$$

with c_M a constant. This is a solution provided

$$g(x,y,t) = c_M e^{-\lambda t} \left\{ \begin{aligned}&\left(D_{11}p_x + D_{12}p_y - up\right)_x \\ &+\left(D_{12}p_x + D_{22}p_y - vp\right)_y\end{aligned} \right\} \tag{4}$$

To attain time-independent Dirichlet boundary conditions from (3), we must have p = 0 on the boundary of the domain. Assume the domain is $\Omega = [0, L]^2$ and let

$$p(x,y) = \sin\frac{\pi x}{L} \sin\frac{\pi y}{L}. \tag{5}$$

To evaluate g explicitly, expand (4):

$$g(x,y,t) = c_M e^{-\lambda t}\begin{Bmatrix} D_{11}p_{xx} + D_{22}p_{yy} \\ +2D_{12}p_{xy} - up_x - vp_y \\ +(D_{11})_x p_x + (D_{12})_x p_y \\ +(D_{12})_y p_x + (D_{22})_y p_y - (u_x + v_y)p \end{Bmatrix}. \tag{6}$$

The partial derivatives of p are:

$$\begin{aligned} p_x &= \frac{\pi}{L}\cos\frac{\pi x}{L}\sin\frac{\pi y}{L}, \\ p_y &= \frac{\pi}{L}\sin\frac{\pi x}{L}\cos\frac{\pi y}{L}, \\ p_{xx} &= -\frac{\pi^2}{L^2}p, \\ p_{yy} &= -\frac{\pi^2}{L^2}p, \\ p_{xy} &= \frac{\pi^2}{L^2}\cos\frac{\pi x}{L}\cos\frac{\pi y}{L} \end{aligned} \tag{7}$$

The derivatives of the Dispersion Tensor are obtained from (2):

$$\begin{aligned} |\mathbf{v}|\frac{\partial D_{ij}}{\partial x_k} &= \left\{\alpha_T\left(\mathbf{v}\cdot\frac{\partial \mathbf{v}}{\partial x_k}\right) + |\mathbf{v}|^2\frac{\partial \alpha_T}{\partial x_k}\right\}\delta_{ij} \\ &+(\alpha_L - \alpha_T)\left\{v_i\frac{\partial v_j}{\partial x_k} + v_j\frac{\partial v_i}{\partial x_k} - \frac{v_i v_j}{|\mathbf{v}|^2}\left(\mathbf{v}\cdot\frac{\partial \mathbf{v}}{\partial x_k}\right)\right\} \\ &+v_i v_j\left(\frac{\partial \alpha_L}{\partial x_k} - \frac{\partial \alpha_T}{\partial x_k}\right) \end{aligned} \tag{8}$$

for $k = 1, 2$.

The solution has been constructed to permit spatially-dependent velocity fields and dispersivities. It is convenient to choose the following functions

$$\alpha_L = \left(\frac{L}{10}\right)e^{vL(x+y)/L}, \tag{9}$$

$$\alpha_T = \left(\frac{L}{100}\right)e^{vL(x+y)/L}. \tag{10}$$

Then

$$\frac{\partial \alpha_L}{\partial x_k} = \frac{\nu_L}{L}\alpha_L \tag{11}$$

and

$$\frac{\partial \alpha_T}{\partial x_k} = \frac{\nu_T}{L}\alpha_T. \tag{12}$$

We suggest $\nu_L = 0.231$ and $\nu_T = 0.366$. For $i = 1,2$, the velocity components are chosen to be:

$$v_i = v_i^0 e^{-\mu_i(x+y)/L}\cos(\omega_i t). \tag{13}$$

Then

$$\frac{\partial v_i}{\partial x_k} = -\frac{\mu_i}{L}v_i \tag{14}$$

(no sum implied on right-hand-side). We suggest $\mu_1 = 0.767$, $\mu_2 = 0.536$. Equation (8) then reduces to

$$|\mathbf{v}|\frac{\partial D_{ij}}{\partial x_k} = \left\{\begin{array}{l} \left\{\alpha_T[\nu_T]\,|\mathbf{v}|^2 - \left(\mu_1 v_1^2 + \mu_2 v_2^2\right)\right\}\delta_{ij} \\ +(\alpha_L - \alpha_T)\left[\dfrac{v_i v_j}{|\mathbf{v}|^2}\left(\mu_1 v_1^2 + \mu_2 v_2^2\right) - \left(\mu_i + \mu_j\right)v_i v_j\right] \\ +(\alpha_L \nu_L - \alpha_T \nu_T)v_i v_j \end{array}\right\} / L \tag{15}$$

again, with no sums implied. In summary, to evaluate the source term, use (6) with the auxiliary relations (5), (7), (2), (13), (14), (15). The initial condition is, of course,

$$c(x,y,0) = c_M P(x,y) \tag{16}$$

Values must be supplied for the parameters u_0, v_0, ω_1, ω_2, ϕ, R, L, λ. and the maximum concentration, c_M. Note that both the solution and the source are independent of the porosity and retardation parameters!

Discharge Calculation

Let B be a subdomain of Ω with $\hat{\mathbf{n}}$ the unit outward normal for a point on the boundary ∂B of the subdomain. The instantaneous mass discharge across ∂B is the line integral

$$\dot{M}(t) = \oint_{\partial B}(\mathbf{F}\cdot\hat{\mathbf{n}})\mathrm{d}\ell \tag{17}$$

where $\mathbf{F} = (F_x, F_y)$ is the flux vector (mass per unit time per unit area):

$$\mathbf{F} = -D\nabla c + \mathbf{v}c \tag{18}$$

If B is the rectangle $[a, b] \times [c, d]$, then the line integral reduces to

$$\dot{M}(t) = \dot{M}_B + \dot{M}_T + \dot{M}_L + \dot{M}_R \tag{19}$$

where

$$\dot{M}_B = -\int_a^b \left(F_y\right)\Big|_{y=c} dx, \tag{20}$$

$$\dot{M}_T = +\int_a^b \left(F_y\right)\Big|_{y=d} dx, \tag{21}$$

$$\dot{M}_L = -\int_c^d \left(F_x\right)\Big|_{x=a} dy, \tag{22}$$

$$\dot{M}_R = +\int_c^d \left(F_x\right)\Big|_{x=b} dy. \tag{23}$$

On Ω,

$$F_x = -D_{11}c_x - D_{12}c_y + uc, \tag{24}$$

$$F_y = -D_{12}c_x - D_{22}c_y + vc. \tag{25}$$

Formulas (2), (3), (9), (10), and (13) can be evaluated on the boundary to determine the 'analytic' instantaneous mass discharge (reducing the expression for the analytic discharge to quadratures).

2D Transport Benchmark Solutions - II

P. Knupp
Ecodynamics Research Associates, P.O. Box 9229,
Albuquerque, NM 87119

The following benchmark is designed to test the implementation of the implicit boundary conditions in the SECOTP2D transport code. The governing equation for this benchmark is:

$$D_{11}c_{xx} + D_{22}c_{yy} + 2D_{12}c_{xy} - uc_x - vc_y = \phi R c_t + \phi R \lambda c + g(x,y,t) \tag{1}$$

where

$$|\mathbf{v}|D_{ij} = \alpha_T |\mathbf{v}|^2 \delta_{ij} + (\alpha_L - \alpha_T) v_i v_j. \tag{2}$$

This equation is obtained from the more general equation solved in the transport code by assuming

- The elements of D are spatially independent,
- Free-water molecular diffusion is zero,
- Dual porosity term is off,
- Single-species decay.

One way to construct a solution which has time-independent boundary conditions is to assume the concentration is of the form

$$c(x,y,t) = c_M p(x,y) e^{-\lambda t}, \tag{3}$$

with c_M a constant. This is a solution provided

$$g(x,y,t) = c_M e^{-\lambda t}\left\{D_{11}P_{xx} + D_{12}P_{yy} + 2D_{12}P_{xy} - uP_x - vP_y\right\}. \tag{4}$$

Assume the domain is $\Omega = [0, L]^2$ and let

$$p(x,y) = e^{-\mu(x+y)/L}. \tag{5}$$

The choice

$$\mu = \frac{u+v}{\sum_{i,j} D_{i,j}} L \tag{6}$$

gives $g(x, y, t) = 0$, which makes this test problem easy to implement.
The initial condition is, of course,

$$c(x,y,0) = c_M p(x,y). \tag{7}$$

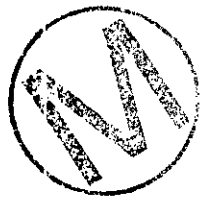
Values must be supplied for the nine parameters α_L, α_T, u, v, ϕ, R, L, λ, and the maximum concentration, c_M. Note that both the solution and the source are independent of the porosity and retardation parameters!

Various boundary conditions can be applied using (3) and (5). For example, to apply a Dirichlet condition to the face $x = 0$, the boundary condition would be

$$c_x(0,y,t) = c_M e^{-\mu y/L} e^{-\lambda t} \tag{8}$$

while the Neumann condition on that face would be

$$c_x(0,y,t) = -\frac{\mu}{L} c_M e^{-\mu y/L} e^{-\lambda t}. \tag{9}$$



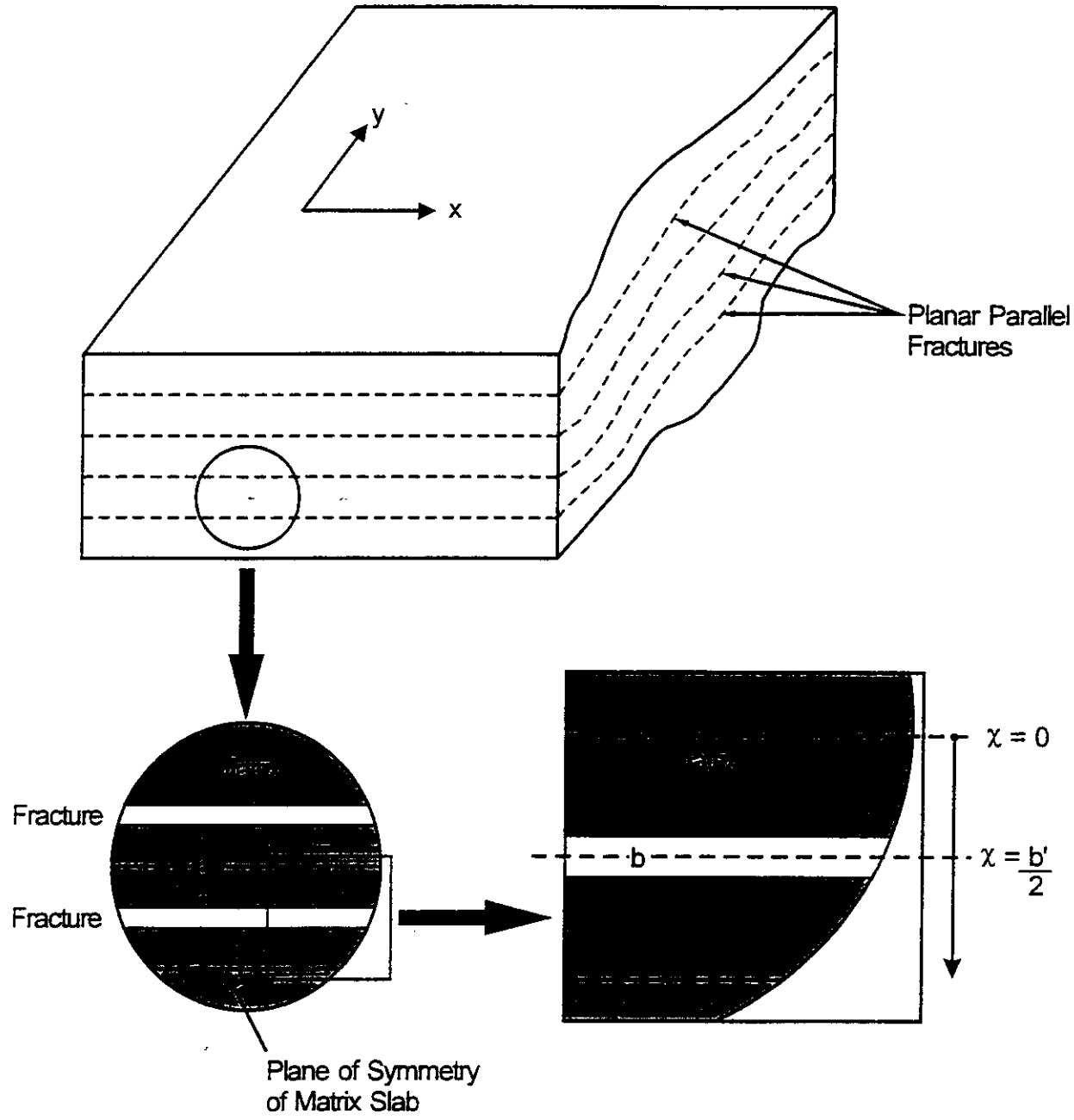
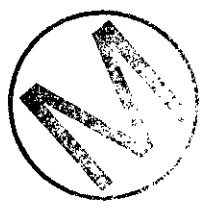


Figure 1 Schematic of dual-porosity model.



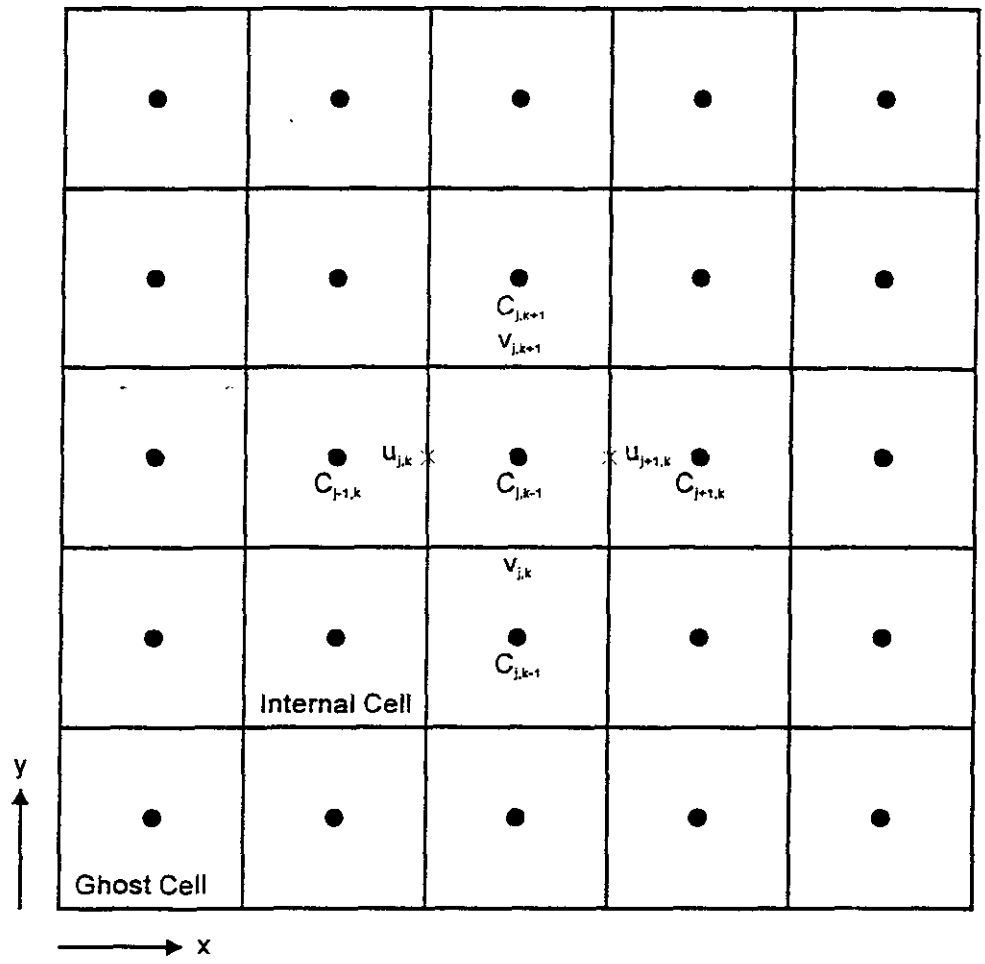


Figure 2 Schematic of finite volume staggered mesh showing internal and ghost cells. The concentrations are defined at cell centers and velocities at cell faces.

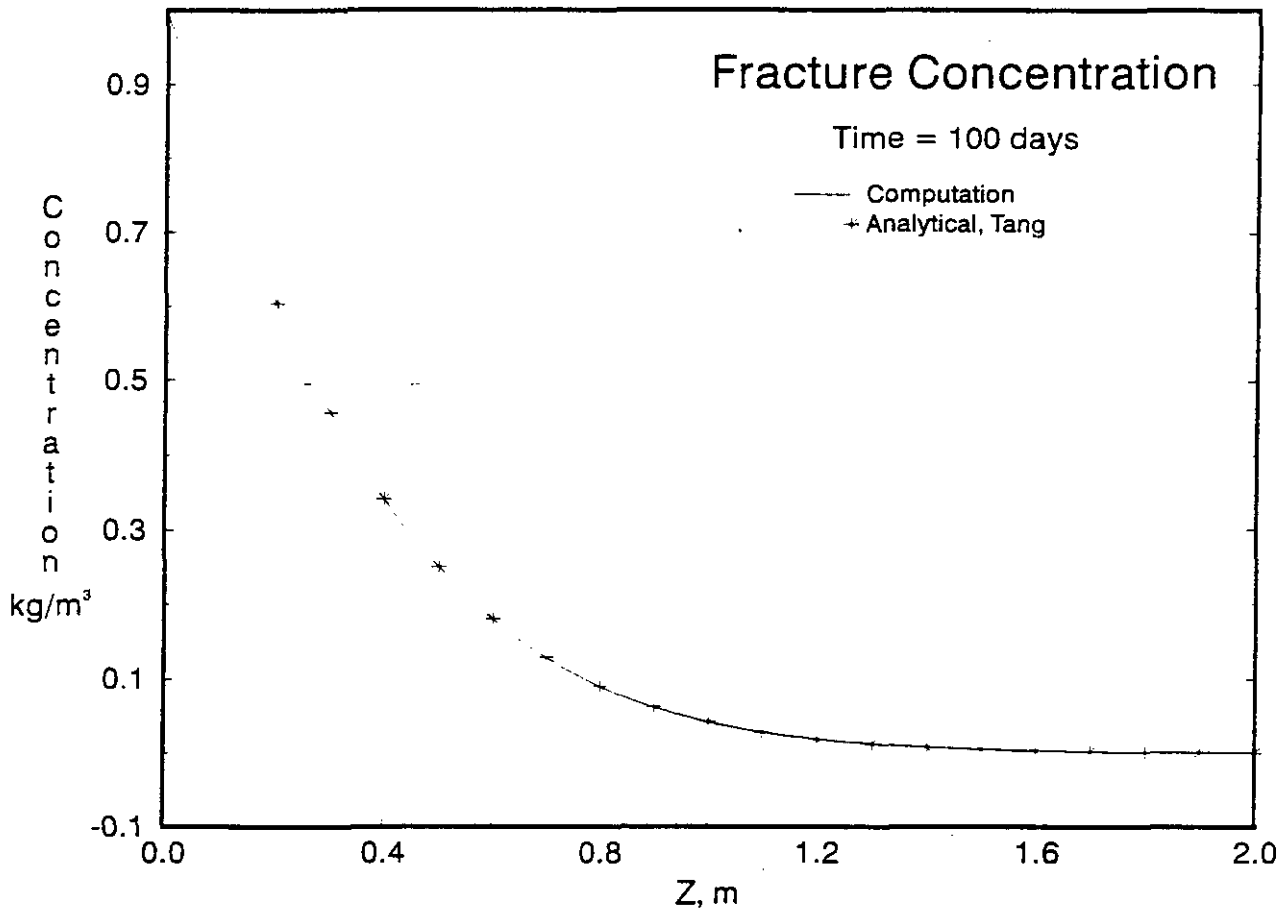


Figure 3 Fracture-Matrix coupling verification: comparison of computed fracture solution to the Tang analytical solution.

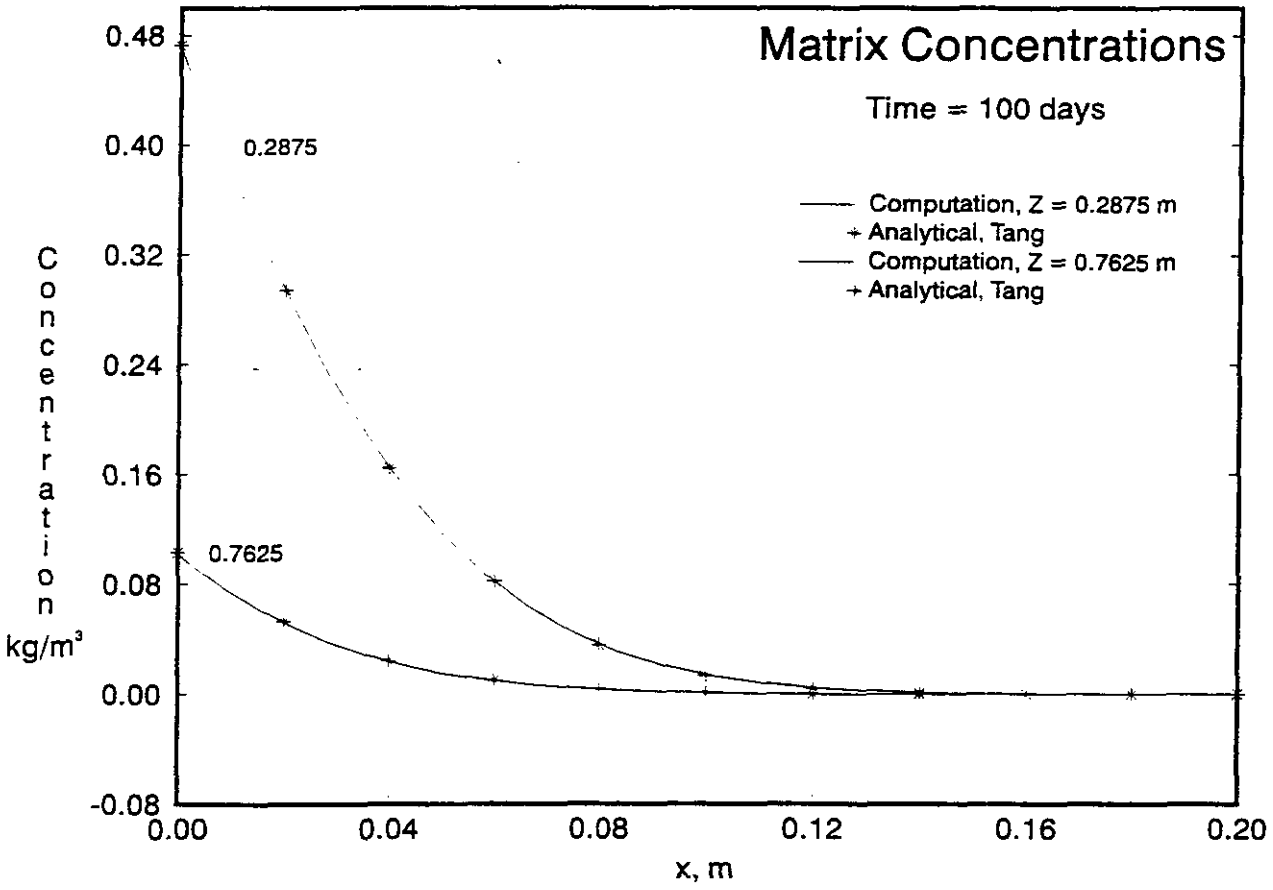


Figure 4 Fracture-Matrix coupling verification: comparison of computed matrix solution to the Tang analytical solution.

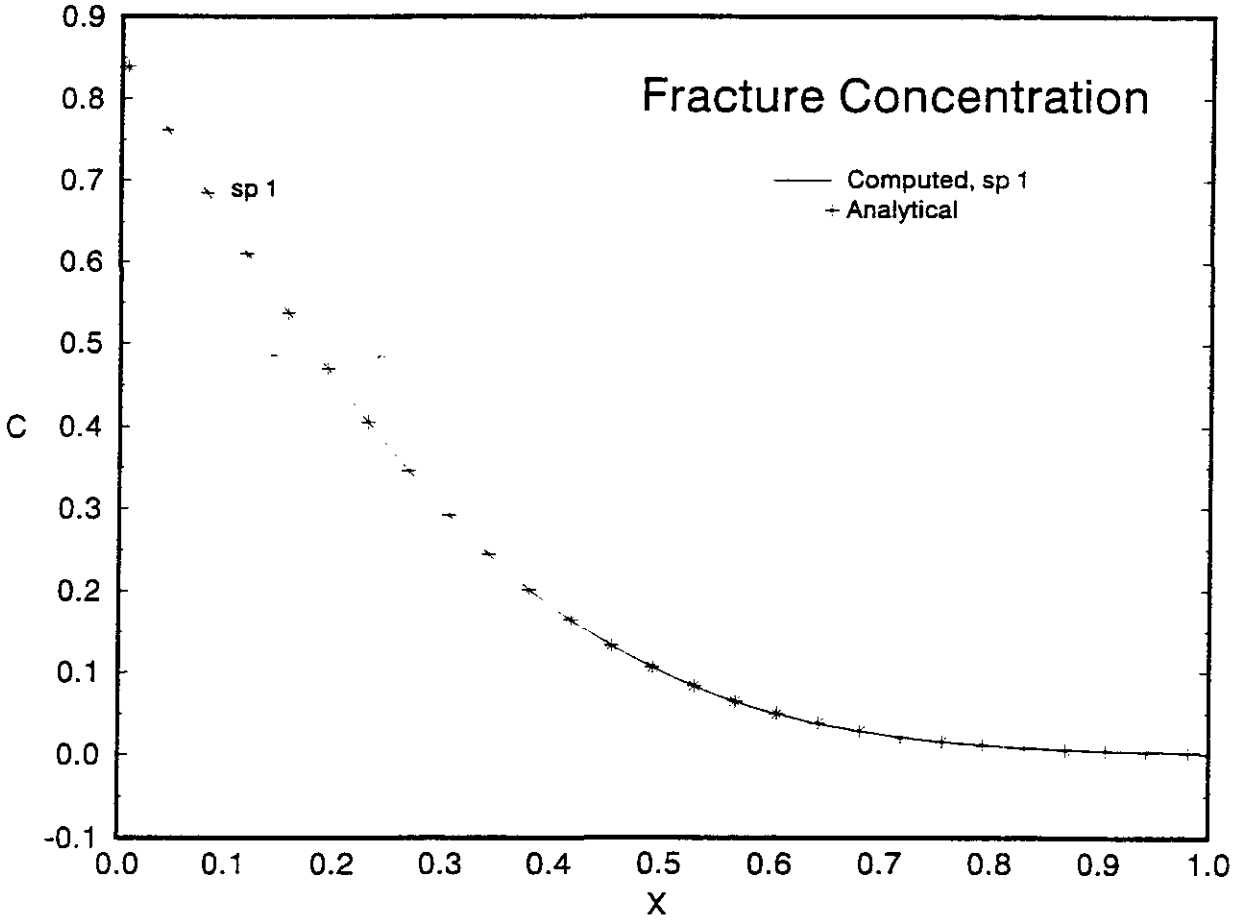


Figure 5 Multiple species verification: comparison of computed fracture solution to the Lester et. al. analytical solution for species 1.

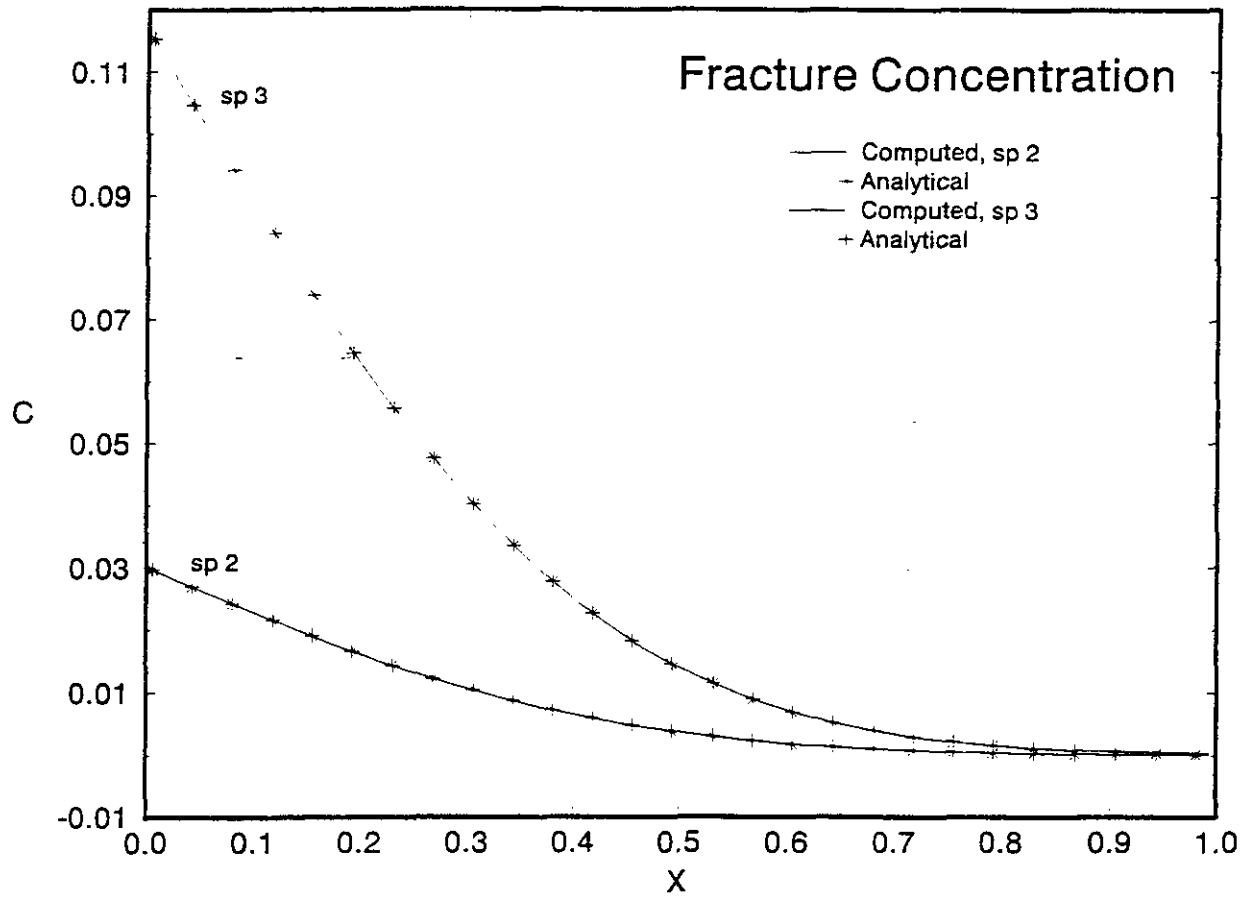


Figure 6 Multiple species verification: comparison of computed fracture solution to the Lester et. al. analytical solution for species 2 and 3.

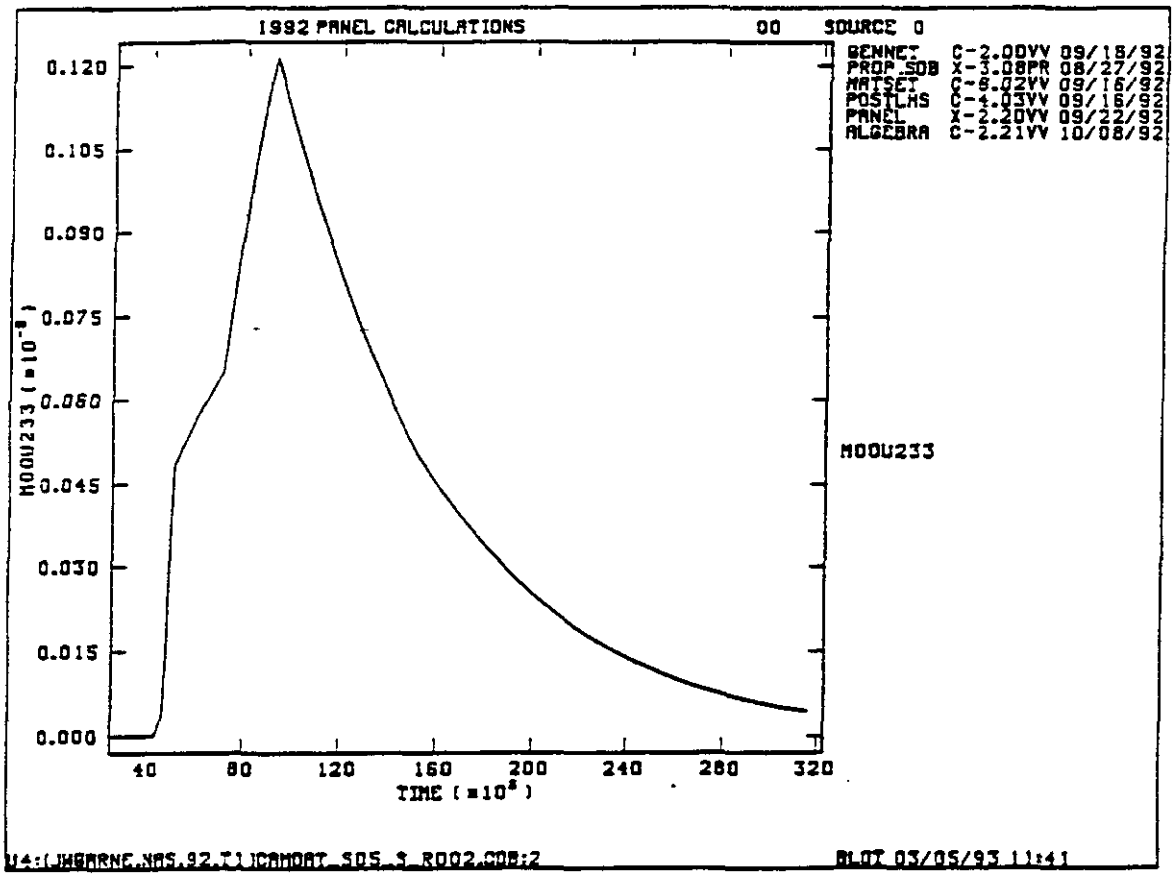


Figure 7 Convergence test on PA problem, vector 2, temporal behavior of the source function.

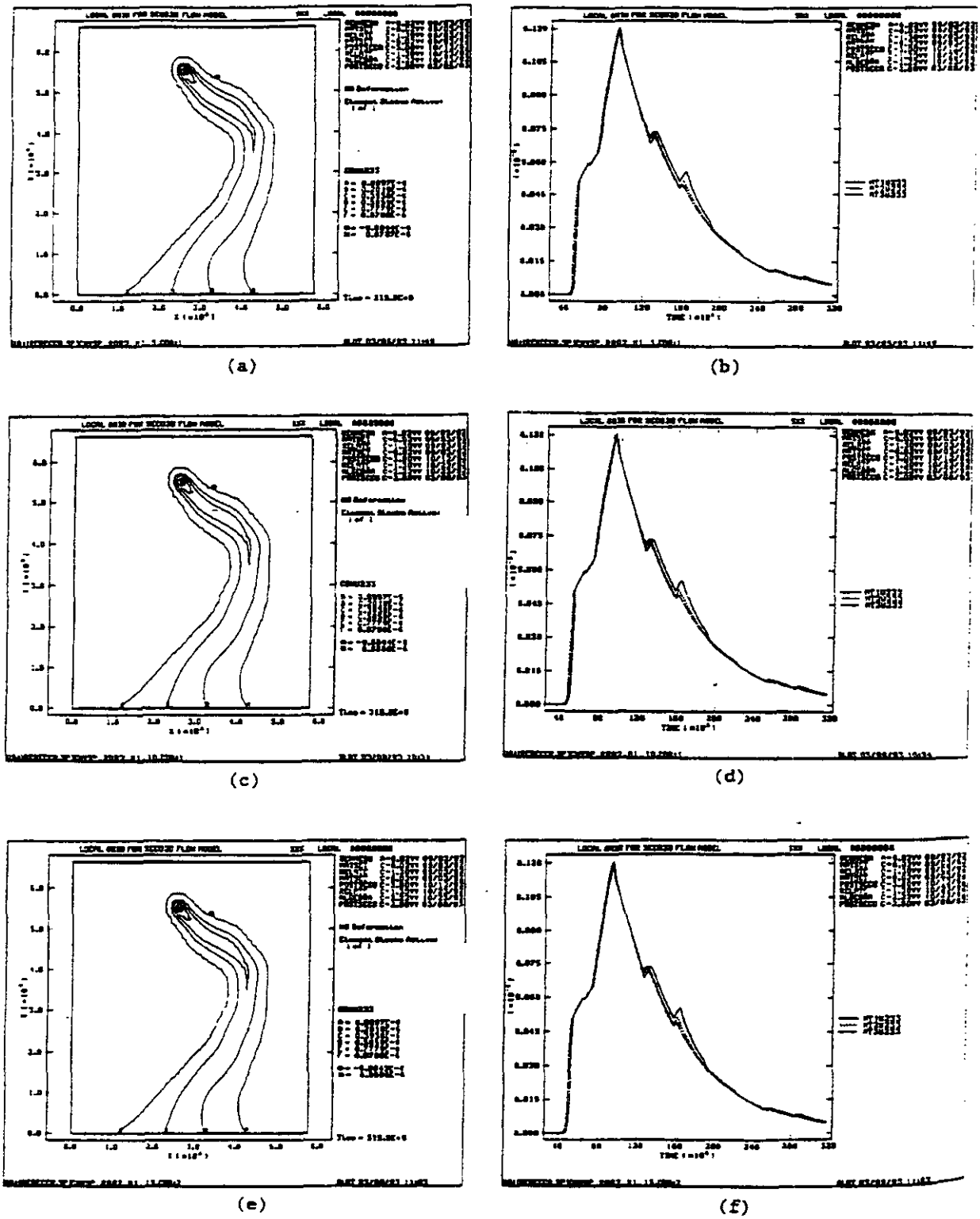


Figure 8 Convergence test on PA problem, vector 2, fracture transport, concentrations contours and breakthrough curves for coarse grid 46x53.

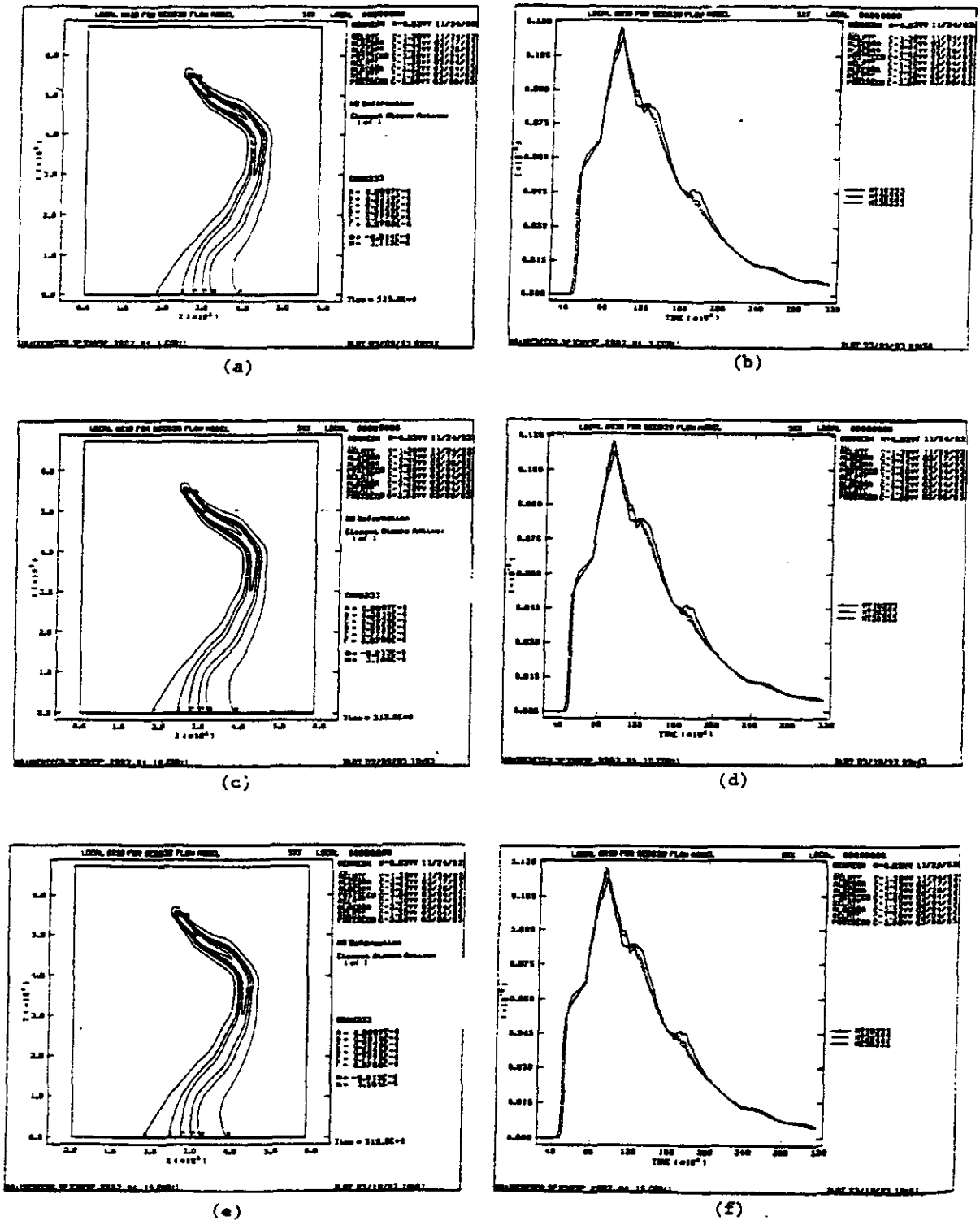


Figure 9 Convergence test on PA problem, vector 2, fracture transport, concentrations contours and breakthrough curves for medium grid 93x107.

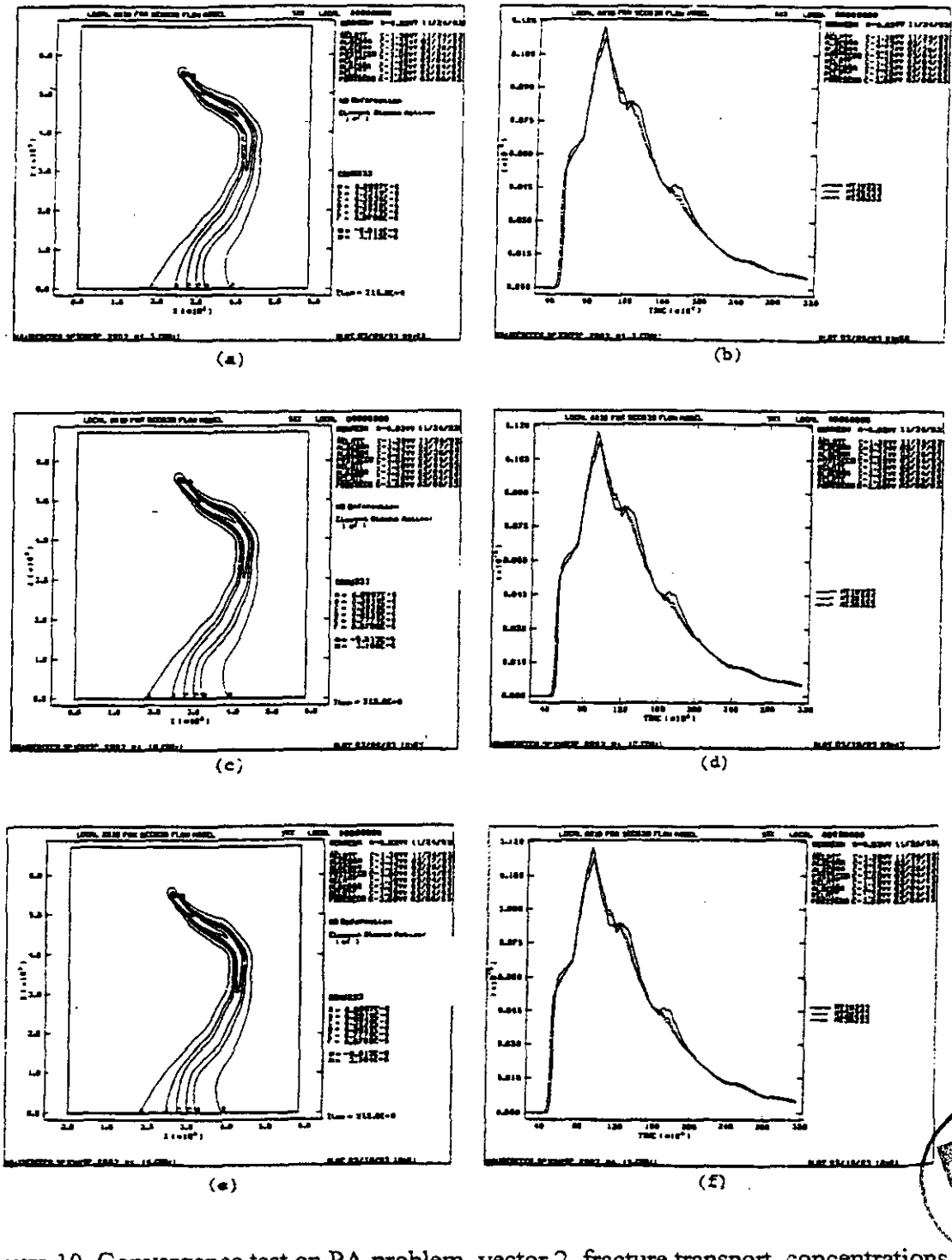


Figure 10 Convergence test on PA problem, vector 2, fracture transport, concentrations contours and breakthrough curves for fine grid 187x215.

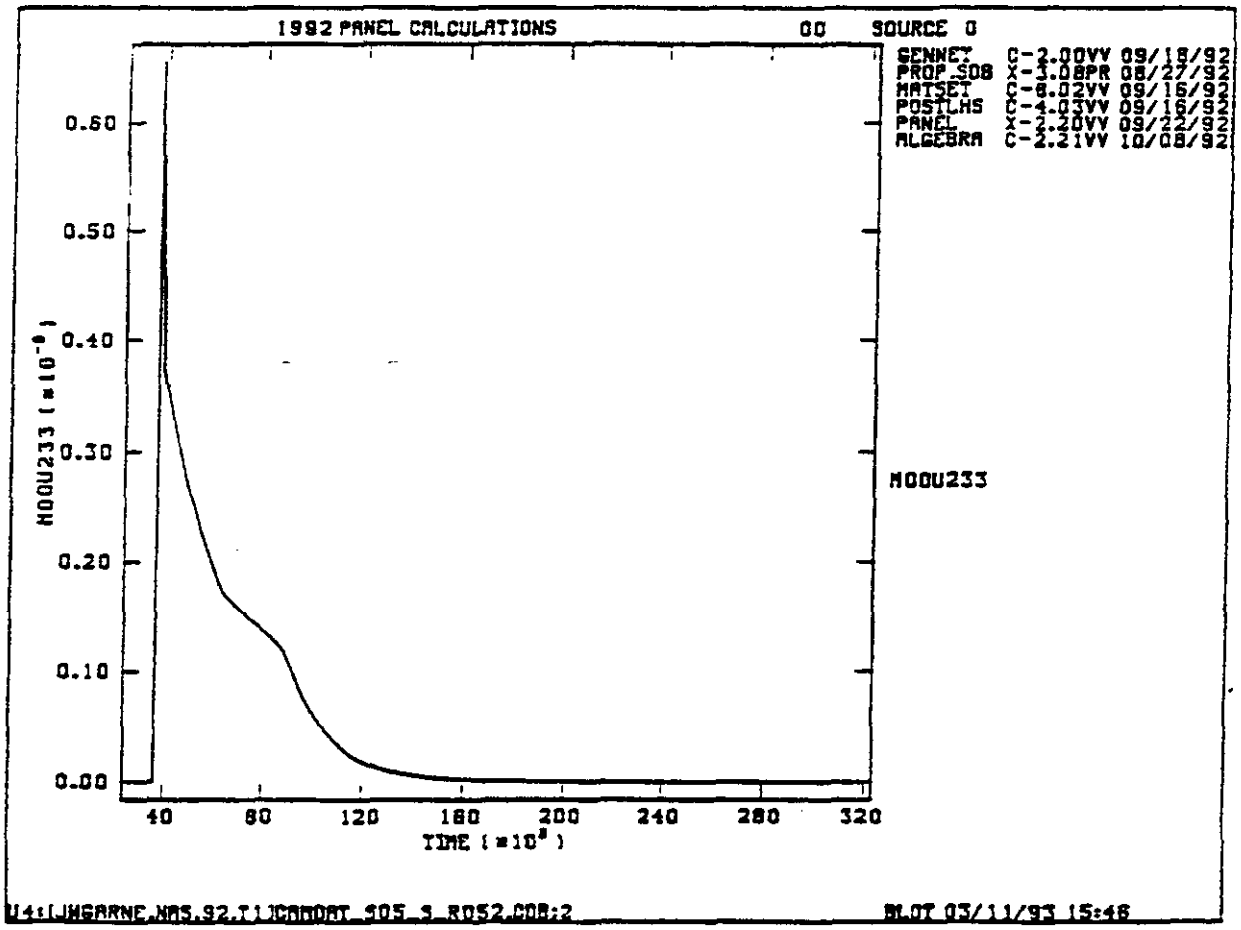


Figure 11 Convergence test on PA problem, vector 52, dual-porosity transport, temporal behavior of the source function.

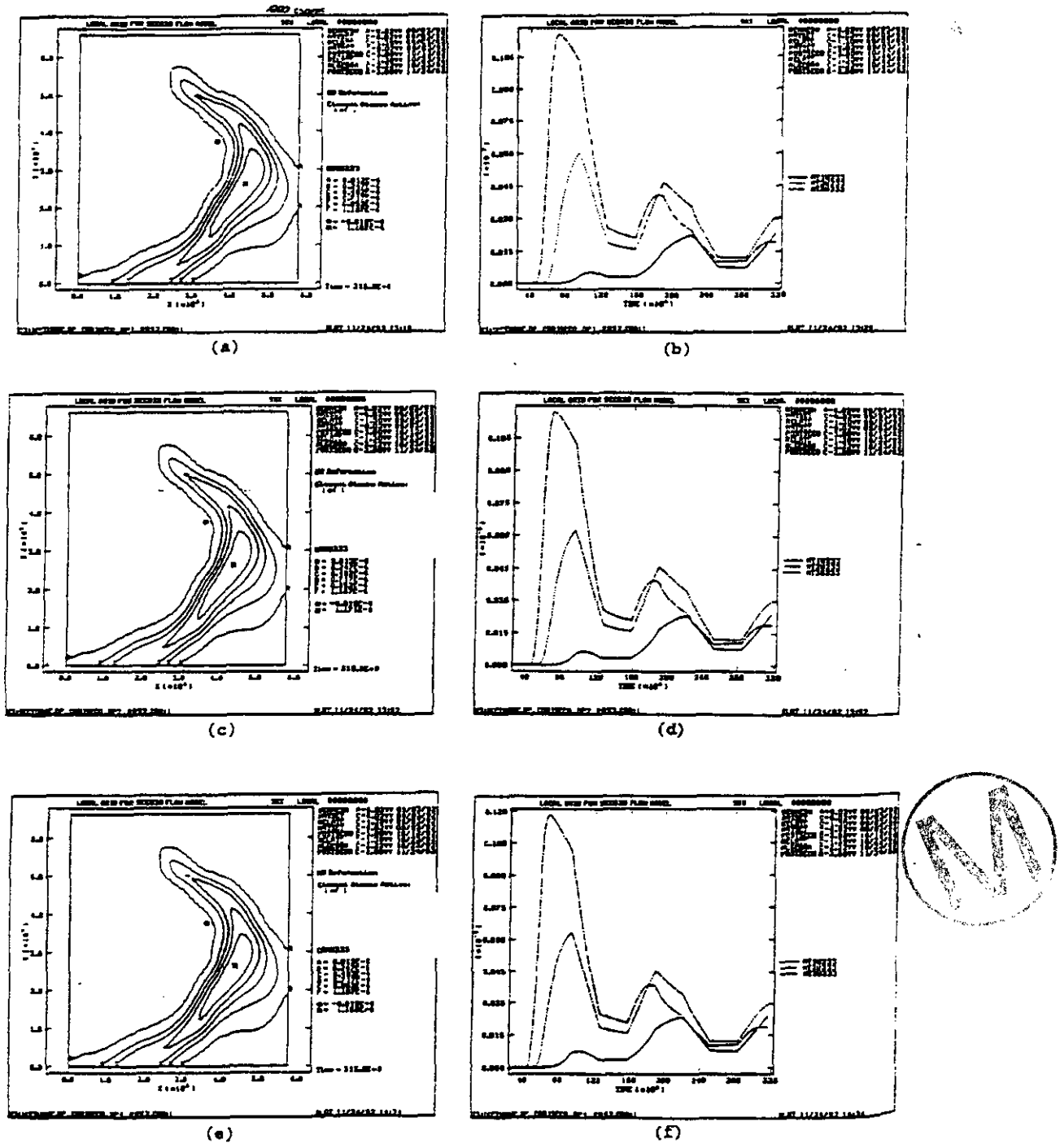


Figure 12 Convergence test on PA problem, vector 52, dual-porosity transport, concentrations contours and breakthrough curves for coarse grid 46x53.

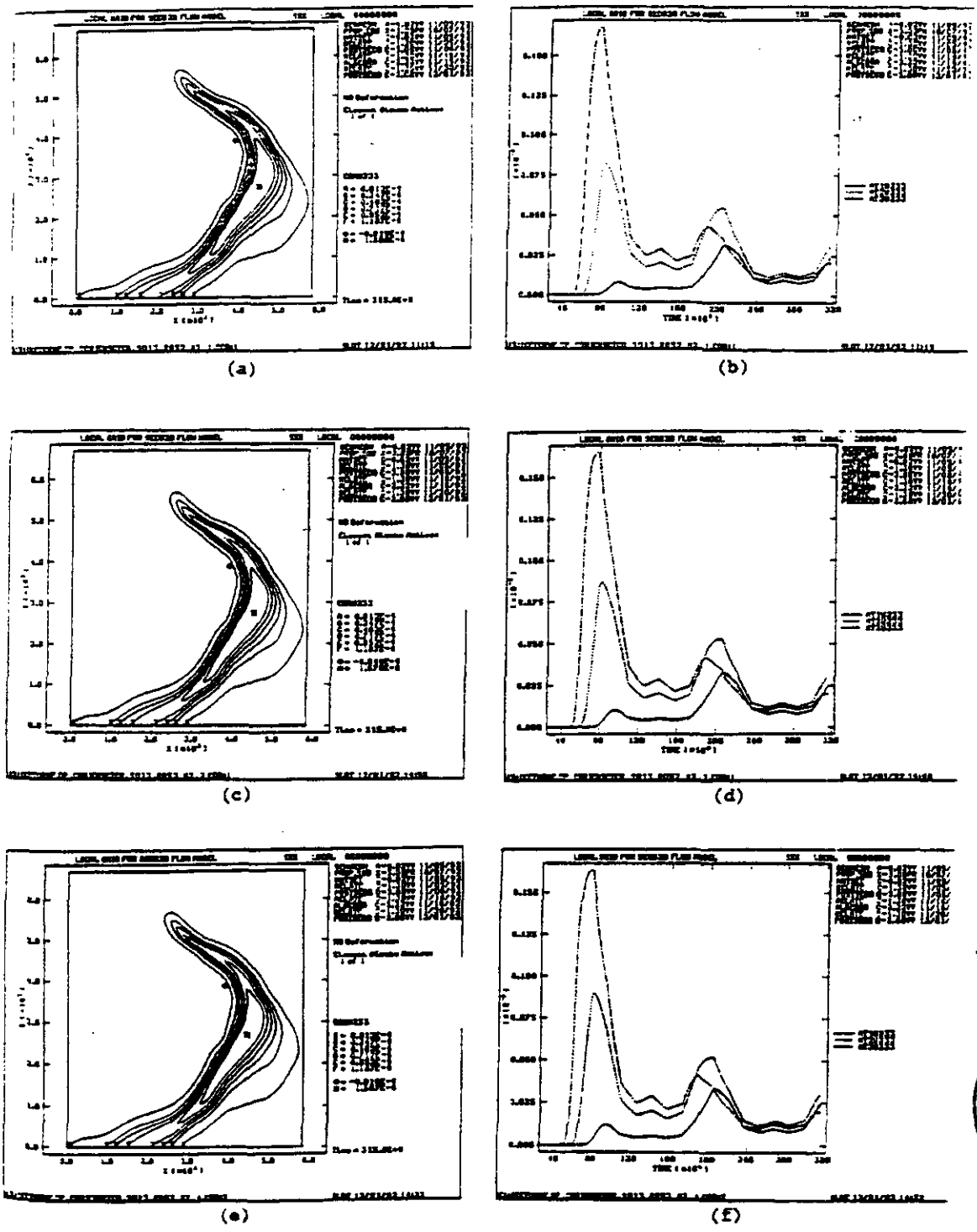


Figure 13 Convergence test on PA problem, vector 52, dual-porosity transport, concentrations contours and breakthrough curves for medium grid 93x107.

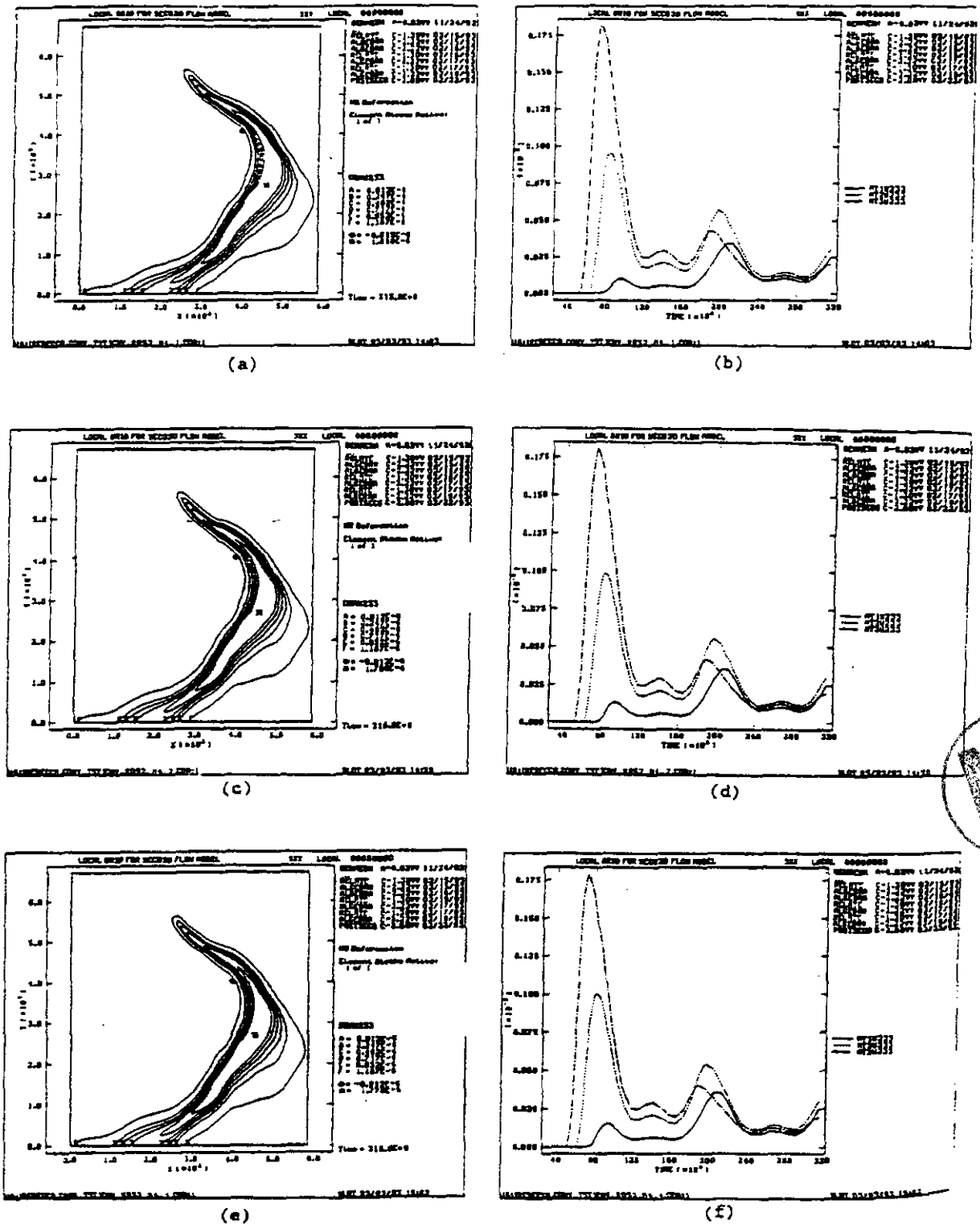


Figure 14 Convergence test on PA problem, vector 52, dual-porosity transport, concentrations contours and breakthrough curves for fine grid 187x215.

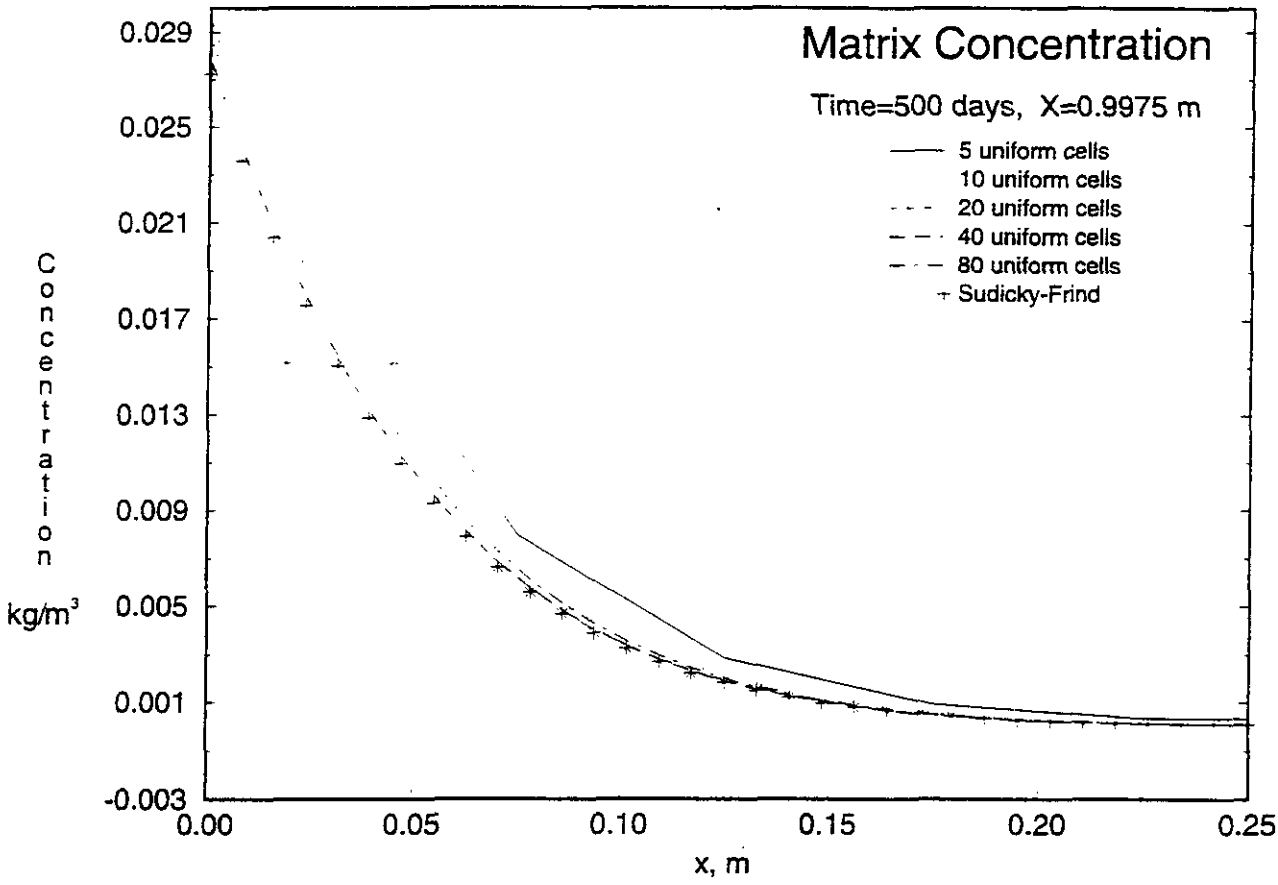


Figure 15 Example problem: Sudicky-Frind, time = 500 days, comparison of grid convergence study to the analytical solution in the matrix.

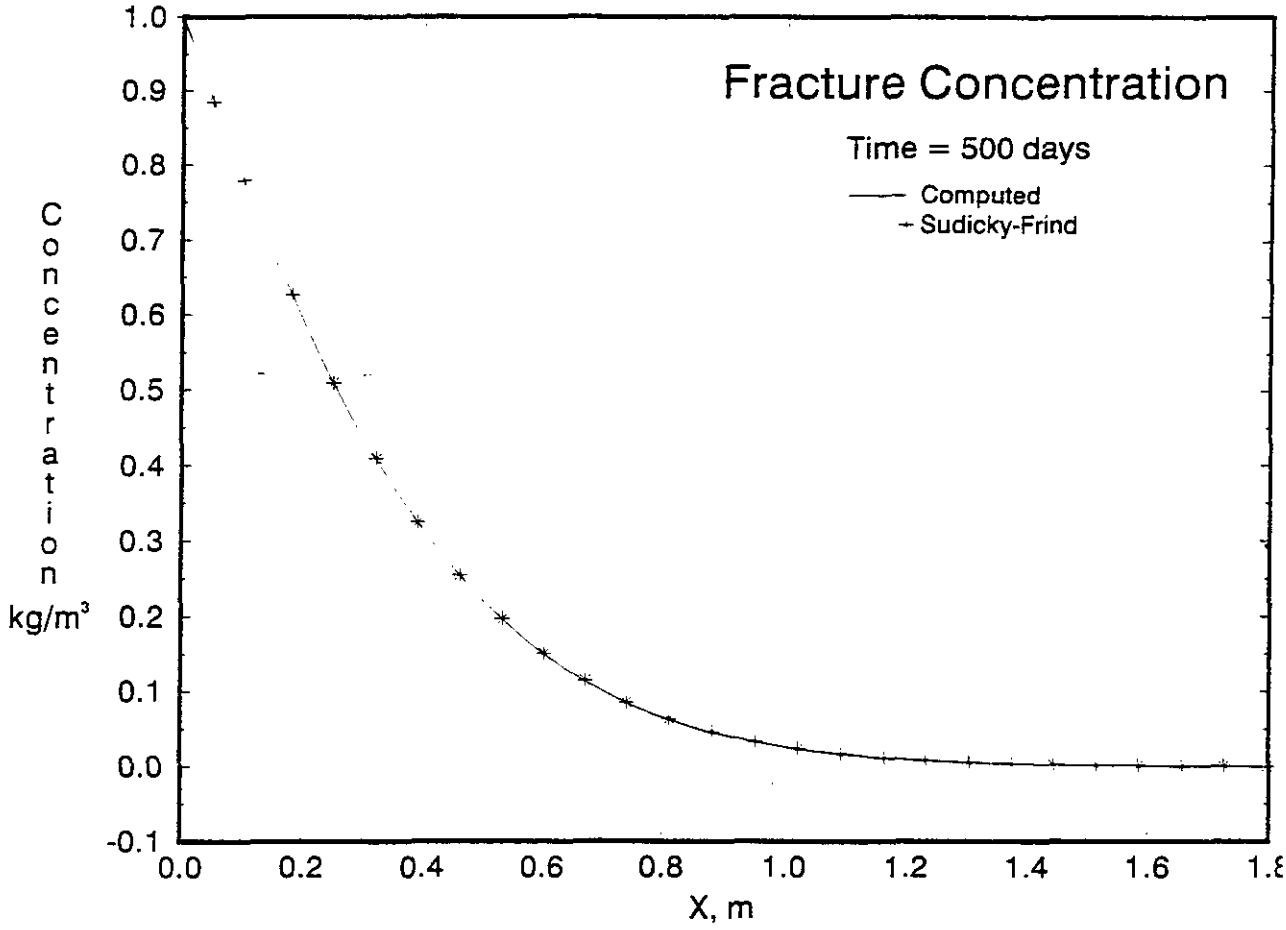


Figure 16 Example problem: Sudicky-Frind, time = 500 days, comparison of computed concentration profile to the analytical solution in the fracture.

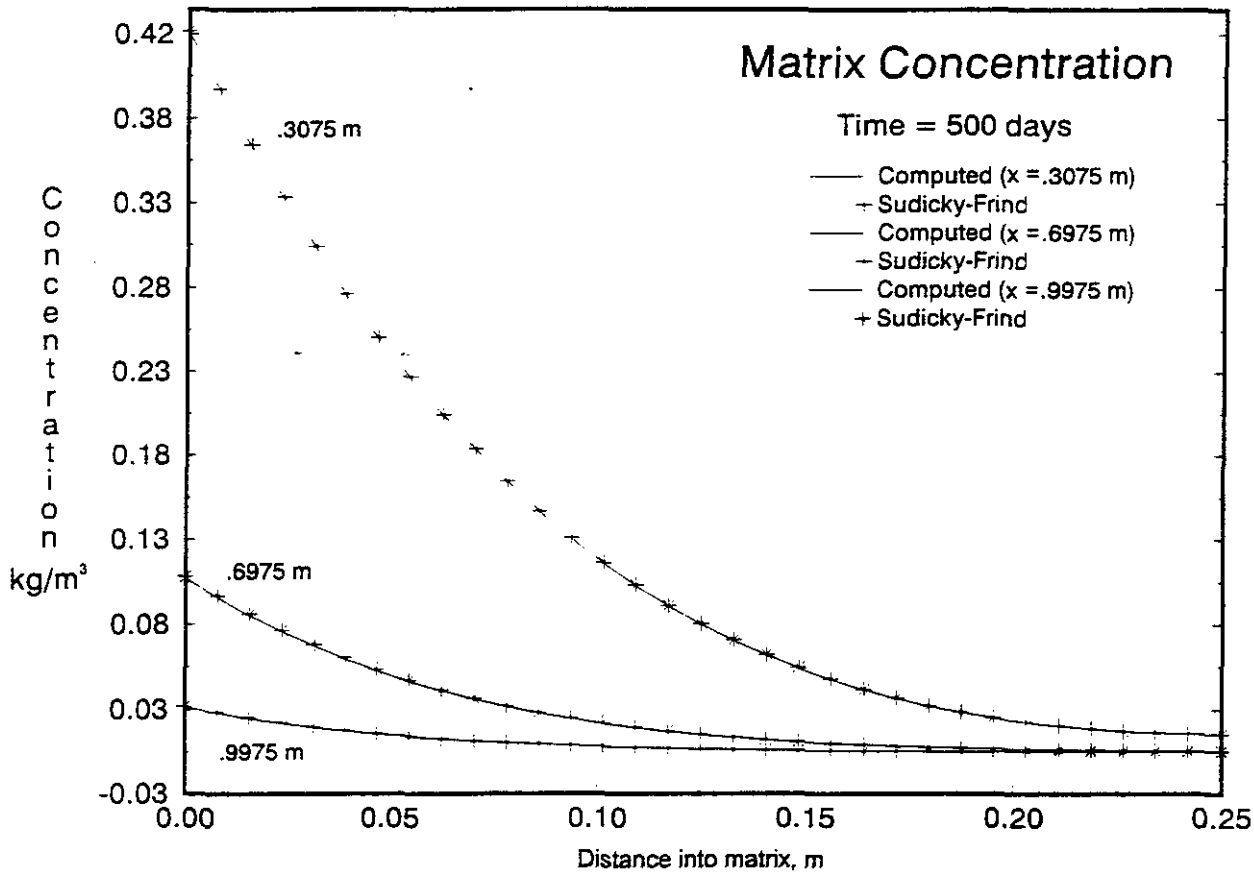


Figure 17 Example problem: Sudicky-Frind, time = 500 days, comparison of computed concentration profile at different fracture locations to the analytical solutions in the matrix.

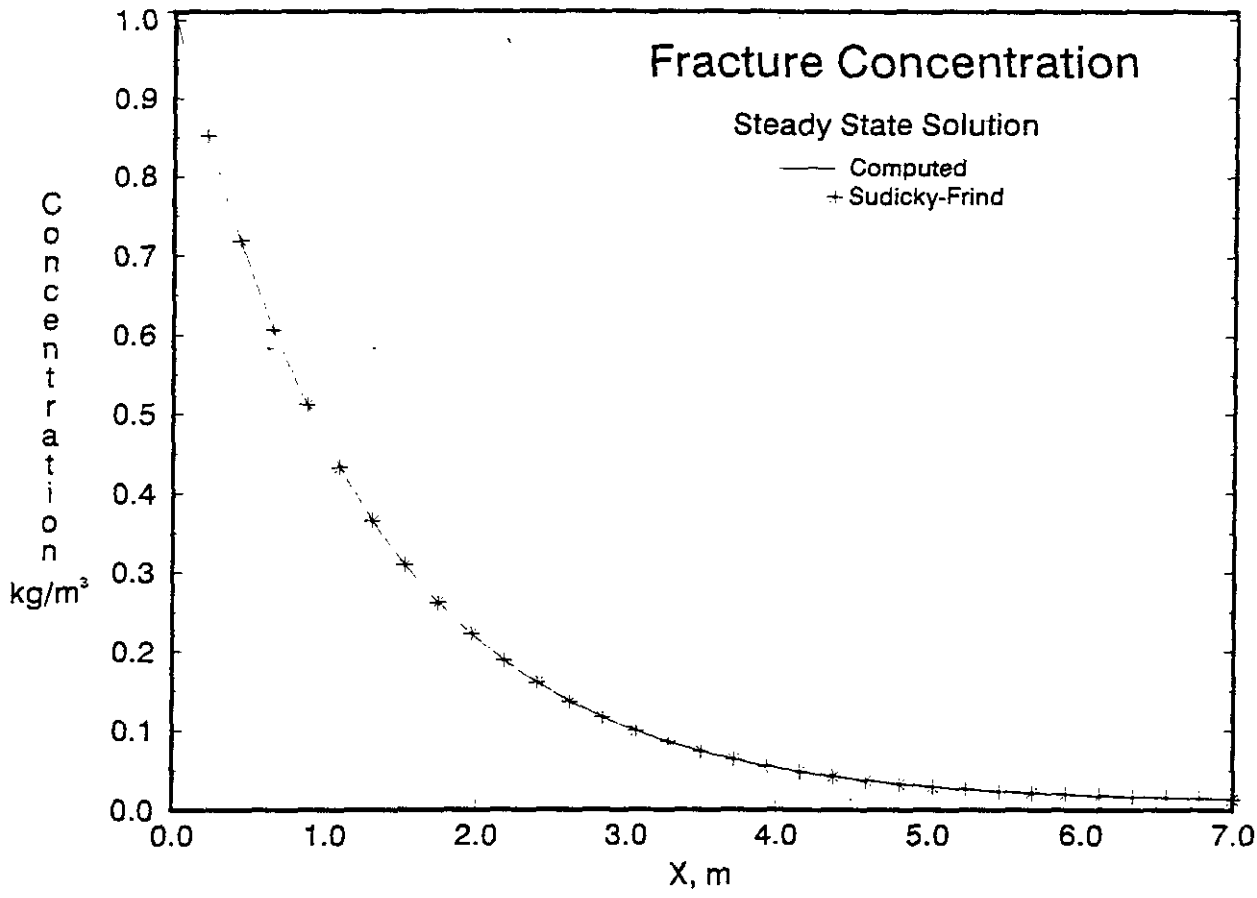


Figure 18 Example problem: Sudicky-Frind, steady state solution, comparison of computed concentration profile to the analytical solution in the fracture.

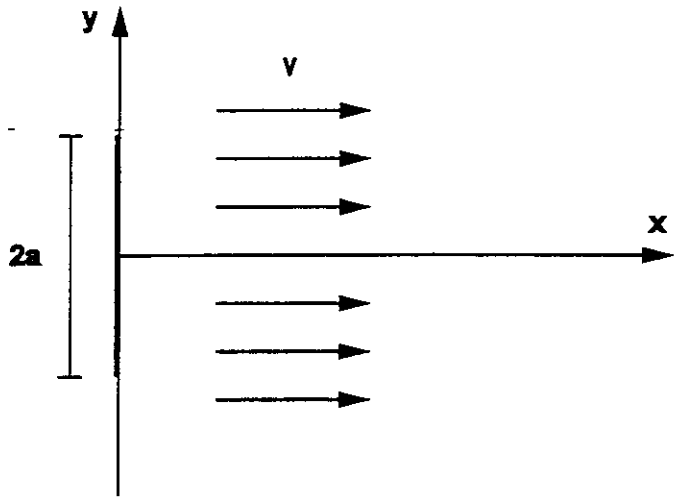


Figure 19 Schematic diagram of 2-D plane dispersion problem.

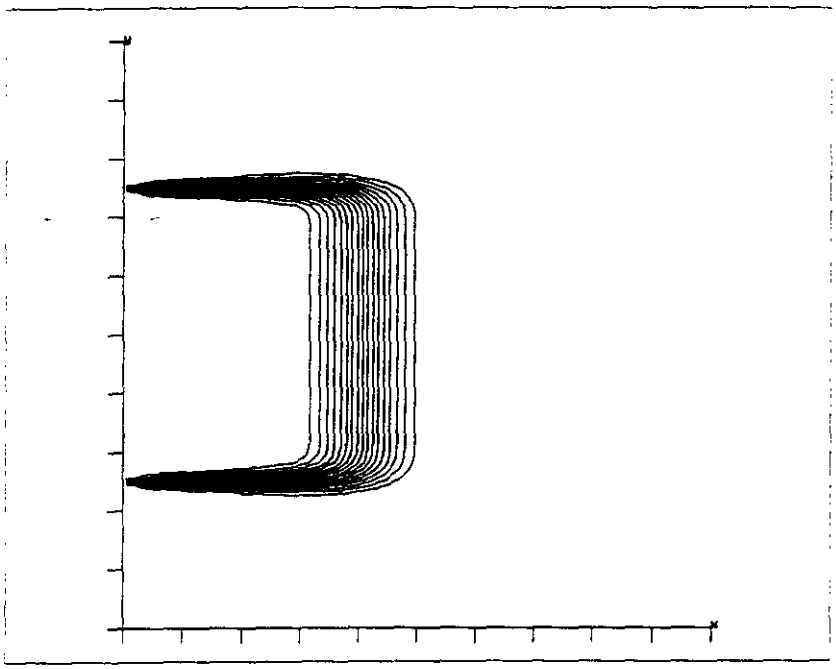


Figure 20 Example problem: fracture transport for low Peclet case, analytical solution, Pe = 2.0.

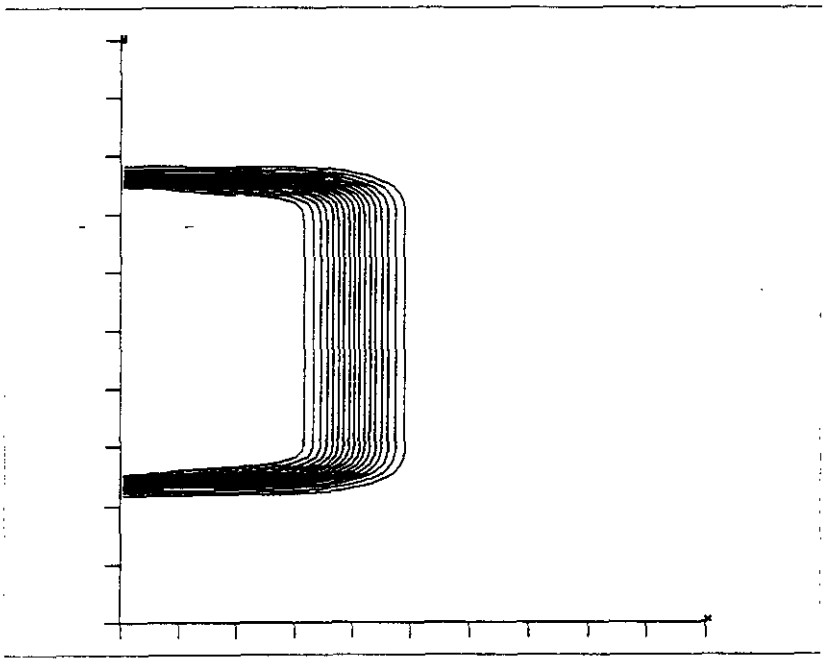


Figure 21 Example problem: fracture transport for low Peclet case, TVD limiter = 7; van Leer's MUSCL limiter, Pe = 2.0.

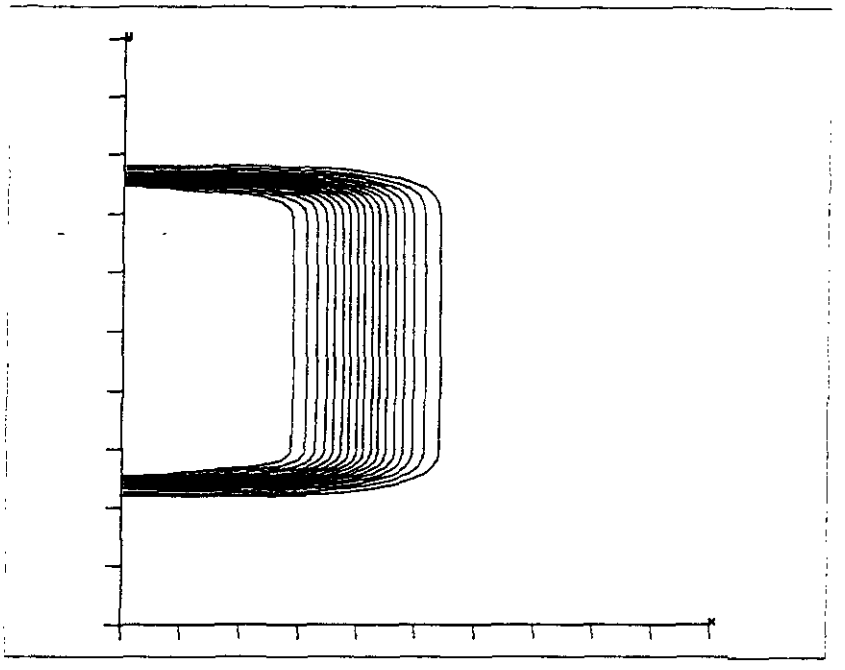


Figure 22 Example problem: fracture transport for low Peclet case, TVD limiter = 1; Upstream differencing, Pe = 2.0.

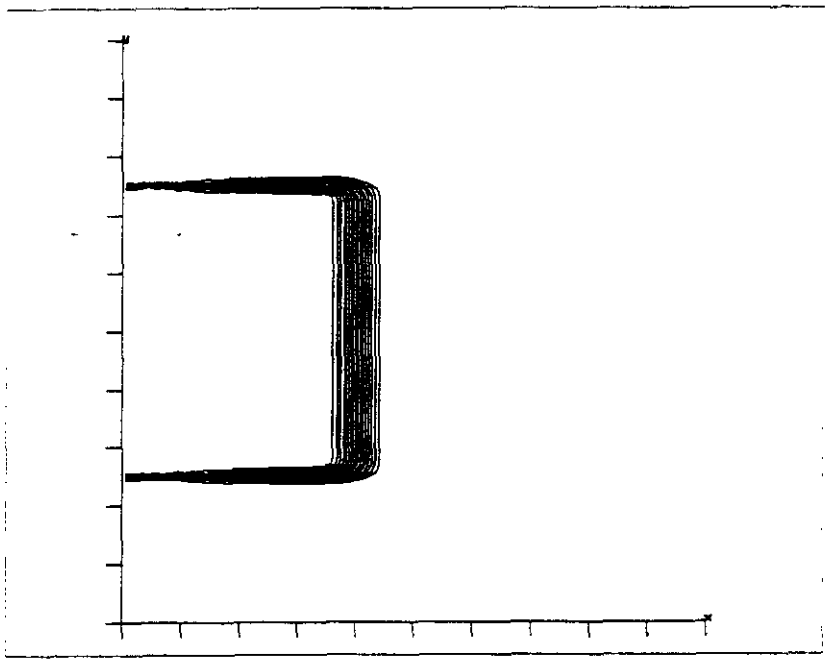


Figure 23 Example problem: fracture transport for high Peclet case, analytical solution, Pe = 10.0.

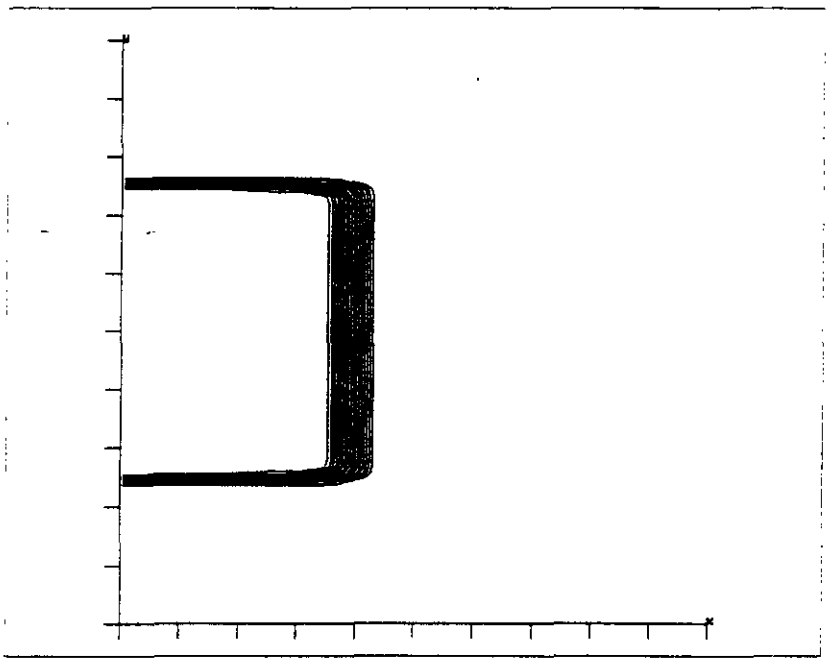


Figure 24 Example problem: fracture transport for high Peclet case, TVD limiter = 7; van Leer's MUSCL limiter, Pe = 10.0.

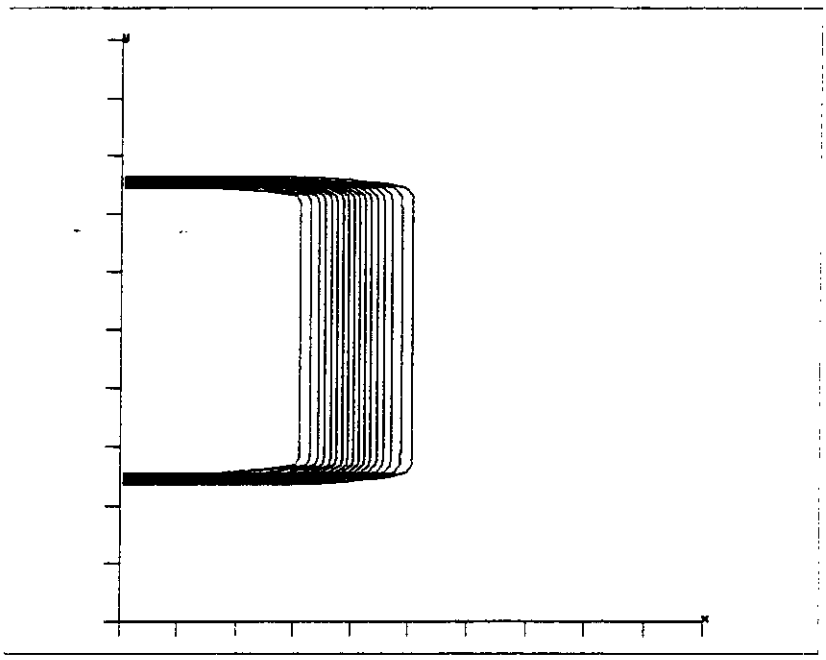


Figure 25 Example problem: fracture transport for high Peclet case, TVD limiter = 1; Upstream differencing, Pe = 10.0.

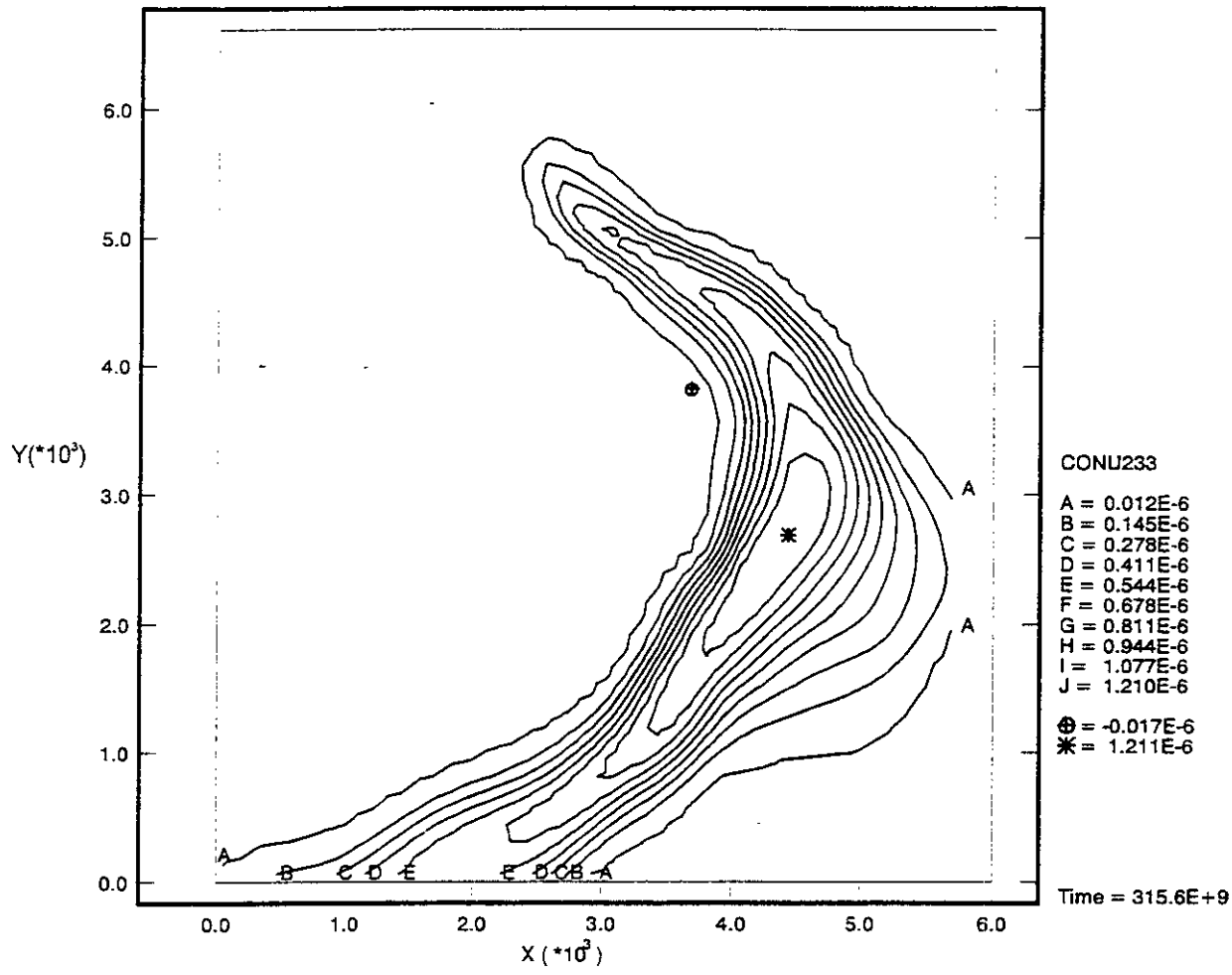


Figure 26 Test problem for SECOTP2D using CAMCON environment, concentration contours of U233.

Appendix II: Sample Diagnostics/Debug File

THE SAMPLE DIAGNOSTICS/DEBUG FILE INCLUDED IN THIS USER'S MANUAL IS NOT THE DIAGNOSTICS/DEBUG FILE PROVIDED IN THE 1996 WIPP PA CALCULATION. IT IS INCLUDED *SOLELY FOR ILLUSTRATIVE PURPOSES.*

```
 SSSSSS   EEEEEEE   CCCCC    OOOOO   TTTTTT   PPPPPP     2222   DDDDDD
SS        EE       CC   CC  OO   OO    TT     PP   PP  2    2  DD   DD
SS        EE       CC       OO   OO    TT     PP   PP       2  DD   DD
 SSSSS    EEEEE    CC       OO   OO    TT     PPPPPP      2    DD   DD
     SS   EE       CC       OO   OO    TT     PP         2     DD   DD
     SS   EE       CC   CC  OO   OO    TT     PP        2      DD   DD
SSSSSS    EEEEEEE   CCCCC    OOOOO     TT     PP       222222  DDDDDD
```

SECOTP2D

SECOTP2D Version 1.21ZO
Version Date 08/16/93
Written by Kambiz Salari
Sponsored by rebecca blaine

Run on 09/22/95 at 14:15:32
Run on ALPHA AXP BEATLE OpenVMS V6.1

SECOTP2D 1.21ZO (08/16/93) 09/22/95 14:15:32

****

Prepared for
Sandia National Laboratories
Albuquerque, New Mexico 87185-5800
for the United States Department of Energy
under Contract DE-AC04-76DP00789

****

Disclaimer

This computer program was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, nor any of their contractors, subcontractors, or their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government, any agency thereof or any of their contractors or subcontractors. The views and opinions

```
          expressed  herein  do not necessarily state or reflect those
          of the United States Government, any agency thereof  or  any
          of their contractors or subcontractors.

          ***************************************************************

*************************************************************************
**

FILE ASSIGNMENTS:
-----------------

    Number of chains  =  1
    Number of species =  1


    Chain:  1     number of species  1
         Species 1 = U233


    Dual-porosity is used
    Number of cells in the block, IMAX =  9


    Number of cells in X-direction, JMAX =  46
    Number of cells in Y-direction, KMAX =  53


    TIME_BEGIN = 3.1557E+10      TIME_END = 3.1557E+11


    Time step is computed, DT = 5.6802E+08


    Type of input velocity: time dependent


    Maximum number of steps, NUMSTEP = 500


    Algorithm parameters:
        Second order implicit 3-point backward time differencing is used.
        Coefficient of source term:   AX = 0.50     AY = 0.50
        Spatial differencing:
           TVD, van Leer MUSCL limiter, LIMITER = 7


    Property and velocity field data files:
        Property file = SECOTP.PRP

        Velocity file = SECOTP.VEL


    Intermediate results are printed at every   50  time steps.
```

```
SECOTP2D CPU time is 1:20 (minute:second)



*** END OF SECOTP2D ***
SECOTP2D 1.21ZO (08/16/93)                                    09/22/95
14:15:32
```

Appendix III: Review Forms

This appendix contains the review forms for the SECOTP2D User's Manual.

NOTE: Copies of the User's Manual Reviewer's Forms are available in the Sandia WIPP Central Files.