A reduced degree of freedom method for simulating non-isothermal multi-phase flow in a porous medium

Peng-Hsiang Tseng *, George A. Zyvoloski

Geoanalysis Group, Los Alamos National Laboratory, EES-5, MS-C306, Los Alamos, NM 87545, USA

Received 22 May 1999; received in revised form 6 December 1999; accepted 19 January 2000

Abstract

Two solution algorithms, with guaranteed memory savings over fully implicit methods, are presented for solving coupled processes in subsurface hydrology. They are applied to problems of non-isothermal multi-phase flow in porous media using both an equivalent continuum approximation and a dual-permeability approach. One algorithm uses an approximation of the Jacobian matrix during the Newton–Raphson iteration such that the coupled system of equations can be partitioned into a solution of some selected primary variables plus a back substitution procedure for the solution of the other variables. The second algorithm uses a similar procedure, except the operations are performed during the preconditioning phase of the solution of linear equations. Numerical forms were derived and the solution procedures were illustrated to reduce the problem from three unknowns per node to one for the single continuum formulation, and from six unknowns per node to two for the dual-permeability approximation. Simulation examples showed that the proposed method produced nearly identical results compared to the traditional fully implicit method while enjoying large memory savings and competitive CPU times.

Keywords: Multi-phase flow; Non-isothermal flow; Dual-permeability model; Reduced degree of freedom method

1. Introduction

Coupled processes are involved in many important engineering and technological problems facing the world today. These problems involve the dynamic interaction of two or more physically distinct subsystems treated as a single system to account for the overall physical effects arising from the coupling. In the areas of flow and transport in porous media, coupled processes represent an abundance of ongoing and new challenges of research for problems such as aquifer restoration, subsurface waste isolation, geothermal reservoir simulation, and enhanced oil recovery. The solution to these problems typically require sophisticated three-dimensional simulations using relevant governing equations that are often highly coupled. The present work was motivated by a need to understand the thermal behavior of a potential nuclear waste repository. To characterize the entire physical system requires not only an integrated understanding of all the coupling processes but also a proper treatment of the solution procedures.

Numerical solution of coupled processes can pose varying degrees of complexity depending on the nature of the processes and their interaction mechanisms. Traditionally, a system of coupled processes has been treated and solved as a single computational entity known as the monolithic augmentation approach [15,30]. Difficulty was soon encountered due to the orders-of-magnitude increases in computational requirements that makes this approach inappropriate for most practical uses. Even for some single-field single-equation solvers the same difficulty remains when a large scale real-world scenario simulation is desired. This has motivated the development of an improved numerical solution method that requires minimal computational effort while maintaining satisfactory numerical stability and accuracy.

In an earlier attempt, Stone and Garder [25] applied a so-called field elimination approach combined with a subsequent explicit updating procedure to solve the coupled one-dimensional equations of gas and oil flow in petroleum reservoir applications. Their method is typically referred to as the implicit pressure-explicit
saturation (IMPES) approach. Although this method avoided the simultaneous solution of coupled equations, the resulting scheme is only conditionally stable. In addition, the many drawbacks of the field elimination approach, as discussed in more detail by Felippa and Park [6] and Park and Felippa [15,16], have greatly restricted its application to other general problems of coupled systems. A type of mixed implicit and explicit scheme was developed, on either a fixed or an adaptive basis, for the solutions of both coupled systems [7,26,27] and single-field problems [12,13,30]. The basic idea behind this type of scheme is that a more efficient explicit formulation can usually be applied to part of the solution domain when the system response in that region is relatively steady. The single-field problem is formulated as a combination of two separate implicit and explicit domains in the context of numerical methods and hence can be defined as a coupled problem in a broader sense [32]. The more appealing adaptive scheme requires a relatively sophisticated bookkeeping process, when the number of coupled equations becomes large, to keep track of the number of nodes to be handled in each domain. For these methods, computer time savings and memory savings are problem dependent. For nonlinear problems, identifying a threshold value to achieve optimum separation of the solution domain depends largely on experience and hence may sometimes overwhelm the advantages of this scheme.

More recently, a method known as the partitioned solution procedure has been pursued and applied successfully to a variety of coupled field problems [5,11,14,17,20,22,23,28,33]. In this approach, the time-integration process is carried out over each suitably decomposed subsystem treated as an isolated entity and recoupled subsequently by appropriate temporal extrapolation techniques. This method greatly enhances computational efficiency and program modularity and has become popular for many practical applications. Unfortunately, the numerical stability and accuracy may be affected by the partitioned treatment and are problem specific. The success of this scheme requires considerable investment in methodology issues to assure a stable and accurate solution. This difficulty has greatly prevented the generalization of this approach to some geoenvironmental applications which may require a solution of six coupled nonlinear equations. An alternative approach, first introduced by Zyvoloski et al. [35] for the applications of nonlinear geothermal problems, circumvents this difficulty while preserving computer memory savings and, to a certain degree, computational efficiency. This approach, while leaving the problem physics unchanged, reduces the degree of freedom at each computational node with some suitable simplifications during the Newton–Raphson iteration approximation. A successive over relaxation (SOR) iteration scheme was later introduced by Zyvoloski [34] as an effective recoupling procedure. This method has the potential to be applied to a number of coupled-field problems, especially when the number of state variables is large.

The objective of this study is to apply the reduced degree of freedom (RDOF) method to the solution of coupled processes involving two-phase flow in a non-isothermal porous medium. Both an equivalent continuum approximation and dual-permeability approach will be considered. For the later case, the solution of six coupled governing equations is required. Simulation examples are presented to illustrate the performance of the reduced degree of freedom method under varying amounts of complexity arising in subsurface flow and energy transport. The simulations were conducted using the Los Alamos National Laboratory code FEHM [36]. The sensitivity of model performance to selected flow parameters is discussed.

2. Mathematical formulation

2.1. Flow and energy-transport equations

For two-phase Darcian flow in a non-isothermal porous medium under the assumptions that viscous heating is negligible, the solid phase is immobile, and local thermal equilibrium between fluid and solid phases holds, the governing equations are taken as [37]

$$\nabla \cdot \left[ (1 - \eta_v) D_{mv} \nabla P_v \right] - \nabla \cdot \left[ (1 - \eta_l) D_{ml} \nabla P_l \right] - \nabla \cdot \left[ \mathcal{Q} \rho_v \nabla (1 - \eta_v) \right] + q_m + \frac{\partial}{\partial z} g \left( 1 - \eta_v D_{mv} \rho_v \right) + (1 - \eta_l) \frac{\partial A_m}{\partial t} = 0, \quad (1a)$$

$$\nabla \cdot \left( D_{el} \nabla P_l \right) - \nabla \cdot \left( D_{ol} \nabla P_l \right) - \nabla \cdot (K \nabla T) + q_e + \frac{\partial}{\partial z} g \left( D_{el} \rho_v + D_{ol} \rho_l \right) + \frac{\partial A_e}{\partial t} = 0, \quad (1b)$$

$$\nabla \cdot \left( \eta_v D_{mv} \nabla P_v \right) - \nabla \cdot \left( \eta_l D_{ml} \nabla P_l \right) - \nabla \cdot \left( \mathcal{Q} \rho_v \nabla \eta_v \right) + q_v + \frac{\partial}{\partial z} g \left( \eta_v D_{mv} \rho_v + \eta_l D_{ml} \rho_l \right) + \frac{\partial A_v}{\partial t} = 0, \quad (1c)$$

where the subscripts v and l denote quantities for the vapor phase and the liquid phase, respectively, P is the phase pressure, T the temperature, K an effective thermal conductivity, g represents the acceleration due to gravity, \( \rho \) the phase density, z the spatial coordinate oriented in the direction of the gravity, t the time, \( \nabla \) the vector differential operator, q the source and sink terms for the corresponding equations, and \( \eta \) is the mass fraction of noncondensable gas in the corresponding phases. By definition, \( \eta_v = \rho_v / \rho_0 \) and \( \eta_l = H_P \rho_0 \), where
tortuosity factor, and \( D \) pore systems to their corresponding bulk properties \[8\].

The mobilities \( D_m \) and \( D_s \) for the mass and energy equations can be expressed as

\[
D_{mv} = \frac{kR_s\rho_s}{\mu_v}, \quad D_{ml} = \frac{kR_s\rho_s}{\mu_l}, \quad (2a)
\]

\[
D_{sv} = h_sD_{mv}, \quad D_{sl} = h_sD_{ml}, \quad (2b)
\]

where \( k \) is the intrinsic permeability, \( \mu \) the viscosity, \( h \) the specific enthalpy, and \( R \) is the relative permeability.

The accumulation terms for Eqs. (1a)–(1c) are given by

\[
A_m = \phi[S_v\rho_v(1 - \eta_v) + S_l\rho_l(1 - \eta_l)], \quad (3a)
\]

\[
A_v = (1 - \phi)\rho_vu_v + \phi(S_v\rho_vu_v + S_l\rho_lu_l), \quad (3b)
\]

\[
A_s = \phi(S_v\rho_v\eta_v + S_l\rho_l\eta_l), \quad (3c)
\]

where \( \phi \) is the porosity, \( S \) the phase saturation, \( u \) the specific internal energy, and the subscript \( s \) refers to the solid phase. Here, \( u_v = c_p T \) where \( c_p \) is the specific heat of the solid material.

The three equations (1a)–(1c), which describe the conservations of mass of water, fluid-medium energy, and mass of noncondensable gas, respectively, may be used to model the flow of air, water, water vapor, and heat in a porous medium. However, in most practical applications the structured nature of a porous medium in field soils or rock formations requires a more complicated bi-continuum approach to describe the behavior of the physical system. One general approach of this type is referred to as the dual-porosity/dual-permeability model or simply dual-permeability model.

2.2. Dual-permeability approach

The dual-permeability model approximates the physical system of a structured medium with two distinct, but interacting, subsystems, one associated with a dominant region of macropores or fractures and the other with a passive region of matrix or porous blocks. Equations which govern flow and energy transport are described separately for each of the two subsystems and coupled by means of a source/sink term to account for the exchange of mass and energy between the two interacting continua. Three parameters are generally used to characterize the two pore systems and their interactions. The first is a volume fraction (of a selected pore system) which relates the material properties of the two pore systems to their corresponding bulk properties \[8\].

The other two parameters, a length scale and a shape factor, are used to quantify interactions between the two pore systems \[1,8\].

The variables and parameters in Eqs. (1a)–(1c) are defined relative to the partial volume of each pore system to account for the relative flow and energy fractions within the two subsystems. Exchange terms describe the transfer of water, water vapor, air, and energy between the two pore systems are defined as

\[
\Gamma_m = \Gamma_{mv} + \Gamma_{ml}, \quad (4a)
\]

\[
\Gamma_s = \Gamma_{sv} + \Gamma_{sl}, \quad (4b)
\]

\[
\Gamma_v = (1 - d_{mv})\Gamma_{mv}h_v^f + d_{mv}\Gamma_{ml}h_v^m + (1 - d_{ml})\Gamma_{ml}h_l^f + d_{ml}\Gamma_{mv}h_l^m + d_{mv}\Gamma_{sv}h_v^s + d_{sv}\Gamma_{sl}h_s^s + \frac{C}{L^2}K(T^f - T^m), \quad (4c)
\]

where

\[
\Gamma_{mv} = \frac{C}{L^2} \frac{k\rho_s\rho_v(1 - \eta_v)}{\mu_v} (P_v^f - P_v^m), \quad (5a)
\]

\[
\Gamma_{ml} = \frac{C}{L^2} \frac{k\rho_s\rho_l\eta_l}{\mu_l} (P_l^f - P_l^m), \quad (5b)
\]

and where the superscripts \( f \) and \( m \) denote the fracture and matrix system, respectively; the quantity \( C \) is a shape factor, and \( L \) represents the distance from the center of a fictitious matrix block to the fracture boundary.

Eq. (4a) specifies that the exchange of water, \( \Gamma_m \), between the two pore systems is the arithmetic sum of two components: the transfer of water vapor and that of liquid water. For positive \( \Gamma_{mv} \) and \( \Gamma_{ml} \), water transfer is directed from the fracture into the matrix in both the vapor and the liquid phases. Similarly, the exchange term \( \Gamma_s \) is the transfer of noncondensable gas in both the vapor phase and that dissolved in the liquid phase. The exchange of energy consists of both a convection and a conduction terms. The convection term is contributed from the transfer of water and noncondensable gas in the two phases. The direction of water or gas flow between the two pore systems is defined as a dimensionless coefficient \( d \) as

\[
d = 0.5 \left( 1 - \frac{\Gamma}{|\Gamma|} \right). \quad (6)
\]

Notice that the subscripts \( mv \), \( ml \), \( sv \), and \( sl \) for \( d \) and \( \Gamma \) are neglected in Eq. (6) for simplicity. For \( d = 0 \) (i.e., positive \( \Gamma \)), the exchange of mass is from fracture to
The fitting coefficients $a_i$ are used in both the numerator and the denominator. 

The pressure-temperature relationships, the two phases (l and v) are given in Zyvoloski et al. [37].

Complete polynomials of degree three of the form

$$a_0 + a_1P + a_2P^2 + a_3P^3 + a_4T + a_5T^2 + a_6T^3 + a_7PT + a_8P^2T + a_9PT^2$$

are used in both the numerator and the denominator. The fitting coefficients $a_i$, ($i = 0 \cdots 9$) for the three physical properties of water (i.e., $\rho$, $h$, and $\mu$) at each of the two phases (l and v) are given in Zyvoloski et al. [37]. The pressure-temperature relationships, $P_i(T)$ and $T(P_i)$, are approximated using rational functions of degree eight [37].

3. The reduced degree of freedom method

The coupled equations (1a)–(1c) were first semi-discretized in space using standard finite element or control volume procedures followed by a generalized one-step time-integration scheme [37]. This resulted in a set of nonlinear algebraic equations written in the form

$$f_i(x_1, x_2, \ldots, x_n) = 0, \quad i = 1, 2, \ldots, n,$$

where each $x_i$, $i = 1, 2, \ldots, n$, denotes a state vector with components $x_i$ through $x_n$, $n$ is the number of state variables per node, and $N$ is the total number of nodes obtained from numerical discretization. Similarly, each $f_i$, $i = 1, 2, \ldots, n$ is a vector with components $f_{1i} \cdots f_{\text{ni}}$. For equivalent continuum model the number of $n$ is 3 and for dual-permeability model $n$ becomes 6. With $N$ nodal points in the finite element discretization, there are $n \times N$ equations and $n \times N$ unknowns to solve.

The resulting computational demand (namely, storage and time) due to this increase in matrix size is obvious. To solve the system of nonlinear equation (9), it is necessary to employ an iterative approach. For some highly nonlinear problems, it is desirable to use the second-order convergent Newton–Raphson method instead of other slower convergent methods such as the Picard iteration [9]. However, this advantage of fast convergence may sometimes be outweighed by the amount of computational operations required to perform each Newton–Raphson iteration. An improved algorithm is introduced as follows to solve Eq. (9) of the coupled processes.

3.1. Equivalent continuum formulation

For most coupled processes, one or more of the processes usually changes slower than the other process. The basic idea behind the reduced degree of freedom method is to pre-factor one or several of those less active degrees of freedom into one or more primary degrees of freedom during the Newton–Raphson approximation. As such, the method has the flexibility to reduce a coupled system of $n$ degrees of freedom per node to a number less than $n$ depending on problem characteristics.

It is important to note that the state variables which dominate a coupled physical process may change with time for certain applications. To enhance the performance of the RDOF method, additional procedures are required to first properly identify the active variables at each time step and to swap the active and the passive variables during the RDOF operations if necessary. The swapping operations can easily be done using a pointer array within the framework of a block matrix structure without influencing the fundamental concept of the RDOF method. The selection of an optimized criterion to best define and identify the active variables, on the other hand, remains largely a rule of thumb. In the following, the method is illustrated to reduce the problem from three unknowns per node to one unknown per node for the single continuum formulation. For the dual-permeability approach, the same procedure is applied to reduce the system from six unknowns per node to two.
unknowns per node, i.e., one unknown for each of the
two pore systems.

The algebraic equation for the Newton–Raphson it-
eration can be written as
\[ J(x_k)(x_{k+1} - x_k) = -f(x_k), \]  
(10)
where \( J \) is the Jacobian matrix, and the subscript \( k \) is the
iteration counter. For the three coupled equations (1a)–
(1c), the resulting Newton–Raphson iteration can be written in block form as
\[ J_{ij}d_{x_j} = -f_i, \quad i, j = 1, 2, 3, \]  
(11)
where \( d_{x_j} = x_{k+1} - x_k \) is the correction vector, \( J_{ij} = \partial f_i/\partial x_j \), and \( i \) and \( j \) represent standard double-
subscript notation for matrices. The iteration counter \( k \) is
neglected for simplicity.

The reduced degree of freedom method is carried out
by an approximation of the Jacobian matrix. In the
following derivation we assume that the variable \( x_1 \) is
the active variable and \( x_2 \) and \( x_3 \) are the passive vari-
ables. Thus in Eq. (11), only the diagonal parts of the
arrays \( J_{22} \) and \( J_{33} \), \( i = 1, 2, 3 \), are retained. For the rest of
the submatrices, the derivatives with respect to the active
variable \( x_1 \), are left in their original form. The solution
of Eq. (11) is now reduced to a system of three de-
coupled equations [35]:
\[ B\delta x_1 = -b, \]  
(12a)
\[ A_1\delta x_2 = -A_2\delta x_1 - a_1, \]  
(12b)
\[ A_1\delta x_3 = -A_3\delta x_1 - a_2, \]  
(12c)
where
\[ A_1 = J_{33}J_{22} - J_{23}J_{32}, \]  
(13a)
\[ A_2 = J_{33}J_{21} - J_{23}J_{31}, \]  
(13b)
\[ A_3 = J_{22}J_{31} - J_{32}J_{21}, \]  
(13c)
\[ B = J_{11} - J_{12}A_1^{-1}A_2 - J_{13}A_1^{-1}A_3, \]  
(13d)
\[ a_1 = J_{33}f_2 - J_{32}f_3, \]  
(13e)
\[ a_2 = J_{32}f_1 - J_{31}f_2, \]  
(13f)
\[ b = f_1 - J_{12}A_1^{-1}a_1 - J_{13}A_1^{-1}a_2. \]  
(13g)

The expressions shown in Eqs. (12a)–(13g) were derived
using the simplified Jacobian matrix. Formally, the de-
coupled equations can be made exact without any simpli-
fying assumptions, that is, even if none of the submatrices are diagonal. However, with the indicated diagonal submatrices, the matrix multiplications and inversions are trivial operations. The matrices \( A_2, A_3, \) and \( B \) are sparse matrices while matrix \( A_1 \) is a diagonal matrix. After the solution for the primary variable \( \delta x_1 \)
using Eq. (12a), the variables \( \delta x_2 \) and \( \delta x_3 \) can be evalu-
ated directly by Eqs. (12b) and (12c), respectively. Note
that the matrix \( B \) has the same sparsity structure as
\( J_{ij}, \ \ i = 1, 2, 3 \), so Eq. (12a) can be solved without extra
storage. Bullivant and Zyvoloski [3] showed that the decom-
position phase (Eqs. (12a)–(13g)) was equivalent to
normalizing the algebraic equations (Eq. (9)) in block
form.

Each iteration of the Newton–Raphson method re-
duores first an evaluation of \( f(x) \) and \( J(x) \), each com-
ponent of which is a scalar function. Since \( f \) has \( 3N \)
components and \( J \) has \( 9N^2 \) elements, there are a total of
\( 9N^2 + 3N \) scalar functions to be evaluated. For the re-
duced degree of freedom method, due to the diagonal
structure assumed in six of the submatrices in \( J \), the
number of scalar functions to be evaluated is reduced to
\( 3N^2 + 9N \)-about one-third that of the original Newton–
Raphson iteration when \( N \) is large.

The solution of the sparse nonsymmetric system (11)
or (12a) can be performed by a standard equation solver
such as the preconditioned conjugate gradient (PCG)
type method. A general step of the PCG method
requires at least one or more matrix–vector multiplica-
tions, various numbers of scalar products and vector–
scalar multiplications depending on the different
versions or variants used [4,21], plus a factorization of
the conditioning matrix. Assume that the number of
nonzero matrix entries is directly proportional to the
order of the matrix. Matrix–vector multiplication
requires \( 3\gamma \times 3N \) operations for the original system of
Eq. (11) where \( \gamma \) is the mean number of nonzero entries
per row of each submatrix \( J_{ij} \). The scalar product
and the vector–scalar multiplication each requires \( 3N \)
operations. For the reduced system of Eqs. (12a)–(12c),
each submatrix has an order of \( N \). The matrix–vector
multiplication requires \( \gamma N \) operations for nondiagonal
matrices and \( N \) for diagonal ones. The scalar product
and the vector–scalar multiplication each requires \( N \)
operations.

In the case of the efficient GMRES algorithm [19], for
example, the average number of operations per step is
\( (m + 2)(3N) + 9\gamma N \) for Eq. (11) where \( m \) is the total
number of steps. The reduced system requires
\( (m + 2)N + \gamma N \) operations to solve Eq. (12a) of order \( N \).
The back substitutions and the calculations to carry out
Eqs. (13a)–(13g) require a total of \( 10N^2 + 6\gamma N \) op-
erations. For the \( LU \) factorization of the conditioning
matrix, the original system requires approximately
\( (3\gamma)^2(3N) \) operations. The reduced system, on the
other hand, requires only \( \gamma^2 N \) operations. One step of
the reduced system thus needs \( m(m + 2) + 10N +
(m + 6)\gamma N + \gamma^2 N \) operations. It can be shown that
the reduced system is always efficient than the original sys-
tem. In fact, the analysis suggests that it would be more
desirable to use the reduced degree of freedom method
when the system becomes large.

The computational efficiency depends also upon the
number of Newton–Raphson iterations and the average
time step sizes required to achieve satisfactory accuracy. Since the Jacobian matrix is altered for the reduced system, in general the solution requires more iterations. Experience has shown that efficiency can be greatly improved when the decoupled system is combined with a recoupling phase of SOR cycles [3,34]

\[
\left[ \frac{1}{\omega} D - L \right] \left[ \frac{1}{\omega} D \right]^{-1} \left[ \frac{1}{\omega} D - U \right] \delta x = -f
\]

(14)

in which the matrix \( J \) of the equation system (11) was decomposed into \( J = L + D + U \), and the parameter \( \omega \) is a relaxation factor. Note again that no additional storage is required for the operations of Eq. (14).

A comprehensive comparative study [3,34] showed that although the reduced degree of freedom method required generally more iterations than solving the original system, the increased number of iteration was only moderate. Hence the overall CPU time used was still far less. Previous studies revealed that in a number of cases evaluated the reduced system allowed a larger average time step size and produced better balance errors [34]. In some cases tested, the reduced system requires even less iterations [3]. The efficiency is thus greatly enhanced for those cases.

In addition to computational efficiency, the reduced degree of freedom method has the advantage of much less storage demand, which makes this method suitable for performing large real-world simulations on present-day personal computers and workstations. A comparison of the storage requirements is illustrated in Table 1 for selected algorithms and equation solvers. The fully implicit entry in Table 1 denotes the simultaneous solution of the coupled set of equations using fully implicit method. Both the RDOF-NR (Newton–Raphson) and the RDOF-solver (linear equation solver) manipulate the Jacobian matrix to obtain a reduced system as described above. The only difference between these two algorithms is in different stages to perform the reduced system manipulation for the GMRES acceleration procedure. The RDOF-NR is implemented using a reduced degree of freedom GMRES and a reduced degree of freedom preconditioning step. The RDOF-solver, on the other hand, uses the full degree-of-freedom GMRES but implements a reduced degree of freedom preconditioning step. The preconditioner of the RDOF-solver is formed using the simplified Jacobian \( J_{ij}, i,j = 1,2,3 \), with only the diagonal parts of the arrays \( J_{ij} \) and \( J_{ij} \) are retained. Because the RDOF-NR algorithm uses a matched set of preconditioning and GMRES acceleration, the method can easily use existing linear equation solver software packages. The RDOF-solver requires some modification of existing software packages. To the authors knowledge, this paper represents the first time the RDOF-solver algorithm has been described and studied. The storage requirements for the ILU factorization and the GMRES (or other acceleration methods) represent the two largest sources of memory usage in the computer code. Both RDOF methods save a factor of 9 in ILU storage and the RDOF-NR method saves a factor of 3 in the GMRES storage.

### 3.2. Dual-permeability formulation

For dual-permeability approach, Eqs. (1a)–(1c) is written for both the fracture and the matrix domains. The resulting equation for the Newton–Raphson iteration has a similar form as Eq. (11) except that the number of degrees of freedom per node has increased from three to six. Using the same physical system as in the single continuum formulation, Eq. (11) can be rewritten for the two pore systems as

\[
\begin{bmatrix}
J_{ij}^{f} & J_{ij}^{m} \\
J_{ij}^{m} & J_{ij}^{mm}
\end{bmatrix}
\begin{bmatrix}
\delta x_{ij}^{f} \\
\delta x_{ij}^{m}
\end{bmatrix}
= - \begin{bmatrix}
f_{ij}^{f} \\
f_{ij}^{m}
\end{bmatrix}, \quad i,j = 1,2,3.
\]

\[(15)\]

### Table 1

Storage requirements for the three algorithms used

<table>
<thead>
<tr>
<th></th>
<th>Fully implicit</th>
<th>RDOF-solver</th>
<th>RDOF-NR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Equivalent continuum</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILU(1)(^a)</td>
<td>(9 \times NC \times N^b)</td>
<td>(NC \times N)</td>
<td>(NC \times N)</td>
</tr>
<tr>
<td>ILU(2)(^c)</td>
<td>(9 \times 1.4 \times NC \times N^d)</td>
<td>(1.4 \times NC \times N)</td>
<td>(1.4 \times NC \times N)</td>
</tr>
<tr>
<td>GMRES</td>
<td>(3 \times (NR + 1) \times N^e)</td>
<td>(3 \times (NR + 1) \times N)</td>
<td>((NR + 1) \times N)</td>
</tr>
<tr>
<td><strong>Dual-permeability</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ILU(1)</td>
<td>(36 \times NC \times N^b)</td>
<td>(4 \times NC \times N)</td>
<td>(4 \times NC \times N)</td>
</tr>
<tr>
<td>ILU(2)</td>
<td>(36 \times 1.4 \times NC \times N^d)</td>
<td>(4 \times 1.4 \times NC \times N)</td>
<td>(4 \times 1.4 \times NC \times N)</td>
</tr>
<tr>
<td>GMRES</td>
<td>(6 \times (NR + 1) \times N^e)</td>
<td>(6 \times (NR + 1) \times N)</td>
<td>(2 \times (NR + 1) \times N)</td>
</tr>
</tbody>
</table>

\(^a\)ILU(1) refers to an incomplete LU factorization and (1) refers to a matrix where fill-in is allowed on only those terms that exist in the original unfactored matrix.

\(^b\)N: Number of nodes in numerical grid; NC: average number of nodal connections.

\(^c\)ILU(2) refers to a matrix where fill-in is allowed to produce additional terms only when the operations involve the original terms of the matrix.

\(^d\)1.4: Fill-in factor for higher order incomplete factorizations (problem dependent).

\(^e\)NR: Number of retained previous solutions.
Similar to the double-subscript notation, the double-superscript of \( \mathbf{J} \) was used to identify the submatrices that form the Jacobian matrix. For instance, \( J_{ij}^{Ef} = \partial E_f / \partial \mathbf{x}_m \). The submatrices \( J^{Ef} \) and \( J^{Em} \) are sparse matrices while \( J^{Ed} \) and \( J^{mf} \) are diagonal matrices which contain the quantities relevant to the exchange terms between the two pore systems.

In the derivations shown below, we assume one dominant variable \( \mathbf{x}_1 \) in both the fracture and the matrix domains appears first

\[
\begin{bmatrix}
J_{11}^{Ef} & J_{12}^{Ef} & J_{13}^{Ef} & J_{14}^{Em} & J_{15}^{Em} & J_{16}^{Em} \\
J_{12}^{Ef} & J_{11}^{Ef} & J_{13}^{Ef} & J_{12}^{Em} & J_{13}^{Em} & J_{14}^{Em} \\
J_{13}^{Ef} & J_{12}^{Ef} & J_{11}^{Ef} & J_{13}^{Em} & J_{12}^{Em} & J_{15}^{Em} \\
J_{14}^{Em} & J_{15}^{Em} & J_{16}^{Em} & J_{14}^{Ef} & J_{15}^{Ef} & J_{16}^{Ef} \\
J_{15}^{Em} & J_{16}^{Em} & J_{14}^{Em} & J_{15}^{Ef} & J_{16}^{Ef} & J_{14}^{Ef} \\
J_{16}^{Em} & J_{14}^{Em} & J_{15}^{Em} & J_{16}^{Ef} & J_{14}^{Ef} & J_{15}^{Ef}
\end{bmatrix}
\begin{bmatrix}
\delta \mathbf{x}_1^n \\
\delta \mathbf{x}_2^n \\
\delta \mathbf{x}_3^n
\end{bmatrix} = \begin{bmatrix}
f_1^n \\
f_2^n \\
f_3^n
\end{bmatrix}
\]  

Following the same procedures as described above for the single continuum approximation, the derivatives of the mobilities are neglected. The submatrices \( J_{ij}^{Ef} \) and \( J_{ij}^{Em} \) are diagonal for \( j = 2, 3 \). Note that the submatrices \( J_{ij}^{Em} \) and \( J_{ij}^{mf} \) have zero entries for \( i \neq j \). To simplify the notation, we neglect the double-superscript and rewrite the Jacobian matrix in Eq. (16) as

\[
\mathbf{J} = \begin{bmatrix}
J_{11} & J_{14} & J_{12} & \ldots & J_{16} \\
J_{14} & J_{14} & J_{24} & \ldots & J_{26} \\
J_{12} & J_{24} & J_{12} & \ldots & J_{16} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
J_{16} & J_{26} & J_{46} & \ldots & J_{16}
\end{bmatrix}
\]  

The coupled system of \( 6N \) equations can be reduced to a system of \( 2N \) equations for the two dominant state variables \( \delta \mathbf{x}_1^n \) and \( \delta \mathbf{x}_m^n \) specified at the \( N \) finite element nodes. The passive variables \( \delta \mathbf{x}_2^n, \delta \mathbf{x}_3^n, \delta \mathbf{x}_m^0, \) and \( \delta \mathbf{x}_m^n \) are evaluated by back substitution after the dominant variables are obtained. The resulting equations, obtained by substitution and elimination, can be expressed as follows for the two dominant variables:

\[
\begin{bmatrix}
Q_{11} & Q_{12} \\
Q_{21} & Q_{22}
\end{bmatrix}
\begin{bmatrix}
\delta \mathbf{x}_1^n \\
\delta \mathbf{x}_m^n
\end{bmatrix} = \begin{bmatrix}
q_1 \\
q_2
\end{bmatrix}
\]  

where

\[
Q_{11} = J_{11} - J_{12}E_f^{-1}E_5 - J_{13}f_1^{-1}F_3 \\
Q_{12} = J_{14} + J_{12}E_f^{-1}E_6 + J_{13}f_1^{-1}F_6 \\
Q_{21} = J_{41} + J_{45}G_1^{-1}G_3 + J_{46}H_1^{-1}H_5 \\
Q_{22} = J_{44} + J_{45}G_1^{-1}G_6 - J_{46}H_1^{-1}H_6
\]  

\[q_1 = -f_1^e - J_{12}E_f^{-1}(J_{12}E_f^{-1}E_5 - J_{13}f_1^{-1}F_3) - J_{13}f_1^{-1}(F_2F_3^{-1} - F_2F_3^{-1} + F_3f_1^{-1}F_2) \\
q_2 = -f_1^e - J_{45}G_1^{-1}(G_2F_3^{-1} - G_3f_1^{-1}F_2 + G_4f_1^{-1}F_4),
\]

and for the passive variables:

\[
E_f \delta x_2^n = -f_2^e + E_2F_3^{-1} - E_3F_2^{-1} - E_4f_1^{-1} + E_6x_1^n, \\
F_1 \delta x_3^n = F_1f_2^e - f_1^e - F_3F_2^{-1} + E_4f_1^{-1} + F_6x_1^n, \\
G_1 \delta x_2^n = G_3f_2^e - G_2f_1^e - G_4f_1^{-1} + G_5f_1^{-1} - G_6x_1^n, \\
H_1 \delta x_3^n = -H_2f_2^e + H_1f_1^e + H_4f_2^e - f_1^e + H_3f_1^{-1} - H_4x_1^n,
\]

where

\[
E_4 = R_3 - R_2R_1^{-1}R_4, \quad E_2 = J_{32}J_{33}^{-1}, \quad E_3 = R_2R_1^{-1}, \\
E_1 = J_{32}J_{35}^{-1} + R_2R_1^{-1}J_{66}J_{55}^{-1}, \quad E_4 = R_2R_1^{-1}, \\
E_4 = R_5 - R_2R_1^{-1}R_6, \quad E_6 = R_5 - R_2R_1^{-1}R_7
\]  

\[
F_1 = S_1 - S_2S_1^{-1}S_4, \quad F_2 = J_{32}J_{33}^{-1} + S_2S_1^{-1}J_{32}J_{32}^{-1}, \\
F_3 = S_2S_1^{-1}, \quad F_4 = J_{36}J_{66}^{-1} + S_2S_1^{-1}J_{36}J_{66}^{-1}, \\
F_5 = S_3 - S_2S_1^{-1}S_6, \quad F_6 = S_8 - S_2S_1^{-1}S_7
\]  

\[
G_1 = S_1 - S_2S_1^{-1}S_2, \quad G_2 = J_{52}J_{52}^{-1} + S_2S_1^{-1}J_{32}J_{32}^{-1}, \\
G_3 = S_2S_1^{-1}, \quad G_4 = J_{56}J_{66}^{-1} + S_2S_1^{-1}J_{36}J_{66}^{-1}, \\
G_5 = S_6 + S_2S_1^{-1}S_5, \quad G_6 = S_9 + S_2S_1^{-1}S_8
\]  

\[
H_1 = R_1 - R_4R_3^{-1}R_2, \quad H_2 = R_4R_3^{-1}, \\
H_3 = J_{63}J_{33}^{-1} + R_4R_3^{-1}J_{23}J_{33}^{-1}, \\
H_4 = J_{65}J_{55}^{-1} + R_4R_3^{-1}J_{55}J_{55}^{-1}, \\
H_5 = R_6 - R_4R_3^{-1}R_4, \quad H_6 = R_7 + R_4R_3^{-1}R_4
\]

in these equations the following definitions are used:

\[
R_1 = J_{66} - J_{63}J_{36}^{-1}J_{36} - J_{65}J_{56}^{-1}J_{56}, \\
R_2 = J_{32}J_{33}^{-1}J_{36} + J_{32}J_{35}^{-1}J_{56}, \\
R_3 = J_{32} - J_{23}J_{33}^{-1}J_{32} - J_{25}J_{55}^{-1}J_{52}, \\
R_4 = J_{63}J_{33}^{-1}J_{32} + J_{65}J_{55}^{-1}J_{52}, \\
R_5 = J_{21} - J_{22}J_{33}^{-1}J_{31}, \quad R_6 = J_{63}J_{33}^{-1}J_{31}, \\
R_7 = J_{64} - J_{65}J_{55}^{-1}J_{54}, \quad R_8 = J_{35}J_{55}^{-1}J_{54}
\]
It can be seen that all the inverse matrices in Eqs. (21a)–(22b) correspond to diagonal matrices. The resulting matrices require a total of \(40N + 16\gamma N\) operations. The matrices \(Q_i\) and the vectors \(q_i\) require \(20N + 8\gamma N\) operations. To solve Eq. (18) by GMRES [19] and the back substitution of Eq. (20), \((m + 2)(2N) + 4\gamma N\) and \(12(N + \gamma N)\) operations, respectively, are needed. Finally, the LU factorization of the conditioning matrix requires \(8\gamma^2 N\) operations. The total number of operations required for the reduced system is thus \(m(2m + 4) + 72N + (4m + 36)\gamma N + 8\gamma^2 N\). Although one step of the GMRES for the original system requires \((6m + 12)N + 36\gamma N\) operations, which is seemingly not too bad as compared to the reduced system, the LU factorization part of the original system needs a lot more operations \((6\gamma)^2(6N)\) than that required by the reduced system. The storage requirements to reduce a system from six unknowns per node to two are given in Table 1.

### 4. Simulation examples

Simulation examples are presented using both the equivalent continuum approximation and the dual-permeability approach. An idealized two-dimensional vertical profile (Fig. 1) of dimensions 560 m deep and 2000 m wide was designed to represent a simplification of the proposed high level nuclear waste repository site at Yucca Mountain. This example was chosen because of its use in repository design and because of its numerical difficulty. The solution experiences boiling, dryout, and re-wetting of liquid water in both the fractures and rock matrix. To compare the performance of various algorithms tested, simple uniform mesh systems of \(30 \times 30\) and \(60 \times 60\) were used for this purpose. The discretization is indicated in Fig. 1 using the long (for \(30 \times 30\) grid) and the short (for \(60 \times 60\) one) tickmarks along the \(x\) and \(y\) axes. An idealized stratigraphic structure that represents the geologic settings at Yucca Mountain was used. Due to the relatively coarse grid used in the simulations, a single layer in the numerical grid system may contain several geologic layers at the Yucca Mountain field site. In this case, a representative hydraulic property was assigned to a certain numerical layer using the most relevant parameter set chosen from Table 2 [31]. The simulations for the coarse grid system (\(30 \times 30\)) were defined by 14 different hydraulic property sets while for the fine grid system (\(60 \times 60\)) a total of 16 parameter sets were used as shown on the right panel of Fig. 1. The properties assumed to be constant in the simulations are: rock bulk density \(\rho_s = 2000\text{ kg/m}^3\), specific heat of rock \(c_p = 1 \times 10^3\text{ J/kg K}\), and the thermal conductivity of rock \(K = 1.0\text{ W/mK}\). The porosity in the fracture domain was defined as one and the residual liquid saturation \(S_w\) was assumed to be 0.01.

The boundary conditions for water flow were assumed to be a constant infiltration rate of 1 mm/yr at the surface and a water table condition at the bottom boundary. For noncondensible gas flow, a specified air pressure of 0.0985 MPa was assumed at the surface and a no flux condition at the lower boundary. For heat transfer, both the surface and the bottom boundaries were assumed to be a flux-type boundary condition. The magnitude of the heat flux is assumed to be proportional to the geologic layers.
Five observation points corresponding to the node numbers of the coarse grid system were chosen, as shown in Fig. 1, for the purpose of comparing the simulation results. Two RDOF methods that differ only in the stages to implement the RDOF manipulation as discussed in the previous section were implemented. The results were compared to the traditional simultaneous solution method using a fully implicit algorithm. Figs. 2 and 3 show the equivalent continuum modeling results of temperature and fluid saturation, respectively, as a function of time for the coarse grid system. All three algorithms generated essentially the same results as shown in the figures. To enhance comparison, a window which shows the enlarged local responses was given in each figure. Only negligibly small differences in temperature can be found between the results generated by the RDOF-NR and the two other algorithms. The total mass balance errors, expressed as a relative error form, ranged from $5 \times 10^{-7}$ to $3 \times 10^{-10}$ while the total energy balance (relative) errors were in the range of $9 \times 10^{-8}$ to $3 \times 10^{-11}$ for the various algorithms tested. The small conservation errors indicate that all the three algorithms performed equally well in modeling the test problem.

For the dual-permeability model, the results obtained from the three algorithms were also found to be indistinguishable as shown in Figs. 4 and 5. Fig. 4 presents

<table>
<thead>
<tr>
<th>Layer index</th>
<th>Matrix domain</th>
<th>Fracture domain</th>
<th>Exchange term</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$S_p$</td>
<td>$x$ ($1/m$)</td>
<td>$n$</td>
</tr>
<tr>
<td>tcw1</td>
<td>0.13</td>
<td>0.012</td>
<td>1.302</td>
</tr>
<tr>
<td>tcw2</td>
<td>0.13</td>
<td>0.013</td>
<td>1.309</td>
</tr>
<tr>
<td>tcw3</td>
<td>0.33</td>
<td>0.002</td>
<td>1.776</td>
</tr>
<tr>
<td>ptn1</td>
<td>0.10</td>
<td>0.397</td>
<td>1.304</td>
</tr>
<tr>
<td>ptn2</td>
<td>0.14</td>
<td>0.107</td>
<td>2.075</td>
</tr>
<tr>
<td>ptn3</td>
<td>0.17</td>
<td>0.440</td>
<td>1.77</td>
</tr>
<tr>
<td>ptn4</td>
<td>0.10</td>
<td>0.450</td>
<td>1.529</td>
</tr>
<tr>
<td>ptn5</td>
<td>0.10</td>
<td>1.182</td>
<td>1.383</td>
</tr>
<tr>
<td>tsw1</td>
<td>0.11</td>
<td>0.104</td>
<td>1.364</td>
</tr>
<tr>
<td>tsw2</td>
<td>0.04</td>
<td>0.399</td>
<td>1.464</td>
</tr>
<tr>
<td>tsw3</td>
<td>0.06</td>
<td>0.062</td>
<td>1.353</td>
</tr>
<tr>
<td>tsw4</td>
<td>0.18</td>
<td>0.008</td>
<td>1.524</td>
</tr>
<tr>
<td>tsw5</td>
<td>0.08</td>
<td>0.024</td>
<td>1.267</td>
</tr>
<tr>
<td>tsw6</td>
<td>0.32</td>
<td>0.006</td>
<td>1.715</td>
</tr>
<tr>
<td>tsw7</td>
<td>0.50</td>
<td>0.001</td>
<td>1.585</td>
</tr>
<tr>
<td>ch1v</td>
<td>0.04</td>
<td>0.597</td>
<td>1.233</td>
</tr>
<tr>
<td>ch1z</td>
<td>0.36</td>
<td>0.002</td>
<td>1.567</td>
</tr>
<tr>
<td>ch2v</td>
<td>0.06</td>
<td>0.746</td>
<td>1.289</td>
</tr>
<tr>
<td>ch3z</td>
<td>0.20</td>
<td>0.035</td>
<td>1.292</td>
</tr>
<tr>
<td>ch4v</td>
<td>0.04</td>
<td>0.597</td>
<td>1.233</td>
</tr>
<tr>
<td>ch4z</td>
<td>0.33</td>
<td>0.001</td>
<td>1.908</td>
</tr>
<tr>
<td>pp2v</td>
<td>0.07</td>
<td>0.172</td>
<td>1.453</td>
</tr>
<tr>
<td>pp2v</td>
<td>0.18</td>
<td>0.017</td>
<td>1.462</td>
</tr>
<tr>
<td>bf2v</td>
<td>0.07</td>
<td>0.172</td>
<td>1.453</td>
</tr>
<tr>
<td>bf2z</td>
<td>0.18</td>
<td>0.017</td>
<td>1.462</td>
</tr>
</tbody>
</table>

$^a$The permeability of the individual fractures. The bulk permeability of the fracture continua is $k 	imes V_f$.

$^b$Volume fraction of the fracture domain.
the results representing the temperature history in the matrix domain. Although the differences in fluid saturation between the fracture and the matrix domains are large as expected, the differences in temperatures are small. The maximum difference in temperature was found to be $1.1^\circ C$ in the vicinity of the heat source (with higher temperature in the matrix domain) during the early stages of heating. This difference drops to $0.4^\circ C$ after 9 yr and to $0.17^\circ C$ after 100 yr. Fig. 4 gave only the results representing the temperature history in the matrix domain. The mass balance errors were found to be in the range of $6.0 \times 10^{-6}$ to $1.2 \times 10^{-7}$ and the energy balance errors were ranged from $4.1 \times 10^{-7}$ to $1.1 \times 10^{-8}$.

All the above discussions apply to the results obtained using a coarse grid system. Further simulations were conducted using a finer grid to test the performance of the RDOF method. The results obtained with the three algorithms were found to be essentially identical and indistinguishable when plotted in the
The total mass and energy balance errors were also found to be in the same ranges as those discussed above. However, there are some subtle differences in the simulation results obtained using two different grid sizes as depicted in Fig. 6. The finer grid predicted a hotter temperature at the heat source and the region became superheated at an earlier time. On the other hand, near the middle of the domain surface the temperatures obtained from the finer grid were cooler. The primary reason responsible for such differences is the averaging of properties and variables during the spatial and temporal discretization implemented in any numerical algorithms. The fact that there were twomore hydraulic parameter sets used in defining the fine grid layer properties has also influenced the solution. The results, nevertheless, indicate the difficulty and importance in capturing the essence of the physical system during a simulation practice. Notice also the differences between the equivalent continuum and the dual-permeability simulation results shown in the figures. Detailed discussions of the conceptual models and the corresponding system behavior will be presented in a separate study. The present study focuses only on the relative performance of the solution algorithm.
5. Comparison of numerical efficiency

Table 3 summarizes the CPU times for each method applied to the repository heating problem described earlier in this paper. In the example problems, pressure was chosen as the active variable and temperature and saturation were the passive variables. For the dual-permeability simulations, the equations were rearranged so the active variables were the pressure in the fracture and the pressure in the matrix. The other variables were taken as passive. Simulations are divided into four subproblems in increasing numerical difficulty: the equivalent continuum formulation on the coarse grid, the dual-permeability formulation on the coarse grid, the equivalent continuum formulation on the fine grid, and the dual-permeability formulation on the fine grid. The simulations show that all the methods have comparable CPU times for all the simulations and thus are sufficiently robust to solve this difficult problem. Except for the smallest problem, the RDOF-NR method was the slowest of the methods and exhibits the most Newton–Raphson iterations. This is to be expected as the RDOF-NR method uses an approximate Jacobian matrix. The RDOF-solver uses the same Jacobian matrix as the fully implicit method; its approximation is done in the preconditioner stage of the linear equation solver. Because of this, the number of Newton–Raphson iterations for the RDOF-solver and the fully implicit method are very close while the iterations for the linear solver increases for the RDOF-solver method.

While the three solution techniques produced nearly identical solutions, numerical errors in both spatial and temporal discretization was high. Some indication of this is evident from the fact that the maximum temperature differences between the coarse and fine grid was over 20°C. It is likely that if models such as these were used for detailed geochemical calculations, the grid resolution and time step size would have to be improved. To examine this effect on the reduced degree of freedom algorithms, the coarse grid dual-permeability calculations were re-run with a maximum timestep size reduced.

<table>
<thead>
<tr>
<th></th>
<th>Equivalent continuum</th>
<th>Dual-permeability</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fully implicit</td>
<td>RDOF-solver</td>
</tr>
<tr>
<td><strong>Coarse grid</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU seconds</td>
<td>568.51</td>
<td>627.67</td>
</tr>
<tr>
<td>Number of timesteps</td>
<td>528</td>
<td>528</td>
</tr>
<tr>
<td>Total N-R iterations</td>
<td>2489</td>
<td>2500</td>
</tr>
<tr>
<td>Total solver iterations</td>
<td>20392</td>
<td>25363</td>
</tr>
<tr>
<td><strong>Fine grid</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU seconds</td>
<td>9313.62</td>
<td>9379.03</td>
</tr>
<tr>
<td>Number of timesteps</td>
<td>1471</td>
<td>1408</td>
</tr>
<tr>
<td>Total N-R iterations</td>
<td>8042</td>
<td>7825</td>
</tr>
<tr>
<td>Total solver iterations</td>
<td>68975</td>
<td>86434</td>
</tr>
</tbody>
</table>
from 10 to 1 yr. With this change, the relative performance of the algorithms also change. The results are given in Table 4. The RDOF-NR method was about 3% slower than the fully implicit method. The RDOF-solver was 13% faster. The RDOF methods depend on the number of SOR iterations allowed in the re-coupling phase of the solution. This parameter has been studied for the RDOF-NR method in previous papers [3,34]. In this work, a maximum of 20 SOR iterations were allowed. The iterations were stopped, however, if the tolerance for the solution of the linear equations was achieved. The effect of SOR iterations on the RDOF-solver algorithm has not been investigated. Table 4 shows that decreasing the number of SOR iterations from 5 to 1 for this algorithm saved about 5% CPU time for the dual-permeability/coarse grid problem.

All the algorithms depended on the fill-in level for the preconditioner of the linear equation solver. The results presented in Table 3 represent simulations with a fill-in level of two. Changing the fill-in level to one saves about 40% for the LU factorization matrix. However, the iterations required in the linear equation solver usually increase. The effect of this change on the RDOF algorithms is given in Table 4. There was a slight decrease in CPU time for the RDOF-solver method and a slight increase in CPU time for the RDOF-NR method. The fully implicit method also experienced a slight decrease in CPU time. There is also an effect of the prescribed solution tolerance. All problems were initially run with a stopping criteria for the Newton–Raphson iteration which forced the block normalized residuals to be smaller than $1 \times 10^{-5}$. The dual-permeability/coarse grid problem was re-run with a tolerance of $1 \times 10^{-4}$. The balance errors again are extremely small (with no discernable change in the solution) but the RDOF-solver ran about 20% faster compared to the $1 \times 10^{-5}$ tolerance. The fully implicit algorithm ran about 14% faster. These results are problem-dependent and should not be generalized.

The RDOF algorithms are sensitive to the choices of active and passive variables. The best choices for the problems considered in this paper involve pressure as an active variable. In isothermal 2-phase calculations, saturation is often the best choice as the air phase is very often passive. To understand the sensitivity to variable ordering, another coarse grid dual permeability simulation with the RDOF-NR algorithm was run with pressure and saturation of the fractures as the active variables. The results, presented in Table 4, show that this simulation ran about 72% slower than the simulation with fracture pressure and matrix pressure as the active variables. This was significant considering the result of the fully implicit simulation for the same ordering, shown also in Table 4, was not (as expected) sensitive to this ordering. Further runs of the dual-permeability model using the fine grid to simulate a period of 10,000 yr revealed that the pressures remained the best choice as active variables. The performance of the RDOF method improved slightly as the temperatures started to decay gradually. Active variable selection is a topic on ongoing research and we expect additional improvements in the future.

The numerical difficulty of this problem causes many restarted timesteps in all the algorithms considered in this paper. Thus, even for the fully implicit method, optimizing iteration and timestep parameters such as the maximum number of iterations, relative stopping criteria of the linear equation solver (usually 0.01 times the current Newton–Raphson residual), the number of previous approximate solutions retained in the GMRES acceleration, and timestep multiplier would result in considerable CPU savings. Going through all these scenarios would be extremely time-consuming and add little to the value of the current paper. Except where noted, the parameters used were identical in all the algorithms used. In addition to the usual iteration parameters, there is a relatively small sensitivity to the over-relaxation parameter for the SOR re-coupling phase. A value of 1.1 was used for the RDOF-NR algorithm and 1.0 for the RDOF-solver algorithm. The sensitivity to this parameter was not studied.

### Table 4
The CPU times used for the dual-permeability model corresponding to the changes of some selected input parameters

<table>
<thead>
<tr>
<th></th>
<th>Fully implicit</th>
<th>RDOF-solver</th>
<th>RDOF-NR</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Coarse grid</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Smaller maximum timestep (1 yr)</td>
<td>3462.33</td>
<td>3044.37</td>
<td>3559.18</td>
</tr>
<tr>
<td>Less SOR iterations</td>
<td>2221.90</td>
<td>1820.33</td>
<td>2019.36</td>
</tr>
<tr>
<td>Larger stopping criteria ($10^{-4}$)</td>
<td>2484.64</td>
<td>2253.87</td>
<td>2642.35</td>
</tr>
<tr>
<td>Change of active variables</td>
<td>2504.63</td>
<td>-</td>
<td>4288.56</td>
</tr>
<tr>
<td><strong>Fine grid</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Larger stopping criteria ($10^{-4}$)</td>
<td>-</td>
<td>28475.16</td>
<td>-</td>
</tr>
<tr>
<td>Less fill-in for preconditioning</td>
<td>-</td>
<td>33004.00</td>
<td>-</td>
</tr>
</tbody>
</table>
6. Remarks

The RDOF methods presented here are especially useful in applications where both flow and contaminant transport is involved. In this case, the timestep is often limited by the transport equations to a smaller value than would be required for the solution of the flow equations only. The smaller timesteps lead to a more diagonally dominant Jacobian matrix and hence a better approximate Jacobian for the RDOF methods.

The linear equation solution methods used in this paper incorporated a very efficient and robust ‘block formation’. Behie and Vinsome [2] describes some of the advantages of this formulation. In this method the elements of the Jacobian matrix represent NVAR by NVAR subblocks and where NVAR is the number of variables for the FIM method or the number of active variables for the RDOF methods. In contrast, using the more common 6N by 6N linear solver for the FIM dual permeability problems would have required 36 times more pointers (pointing to non-zero matrix elements) for the problem than the block formation used in the paper. This is because the block formulation uses an N by N matrix (with 6 by 6 submatrices) and associated non-zero pointers. If the standard non-block linear solvers were used, the RDOF methods could also claim an additional factor of 9 savings for the integer pointer memory. Even though this is ‘integer storage’ its size is equal to that of the Jacobian matrix.

It should also be noted that the block formulations are generally not available for more than several variables. The 6 by 6 block solver was developed at Los Alamos for this application and is a special case. Thus the RDOF methods exhibit a practical advantage of allowing block solution methods to be used when a block method would not be available for the corresponding FIM solution.

We note that RDOF methods offer enough memory savings to allow problems that once were tractable only on workstations to be solved on PCs without swapping out to disk. Though these methods have been tested of a special class of problems, the authors experience has shown them to beneficial for other classes of problems including isothermal simulations.

The results presented here use the GMRES acceleration method. There are other acceleration methods available which may perform better. The BICGSTAB(l) method [24] is very promising. The algorithms described here would benefit from a study of the algorithms when used in conjunction with the BICGSTAB(l) acceleration. In addition, a matrix-free Newton–Krylov method [10] can be potentially combined with the reduced system to further save memory usage.

7. Conclusions

Two algorithms, with significant computer memory savings over traditionally fully implicit methods, are capable of solving some of the most difficult problems in underground flow: dual-permeability simulations of coupled water, air, and energy flow. In contrast to adaptive implicit methods and other implicit–explicit methods, the algorithms presented here are conditionally stable. If the underlying assumptions are violated, the methods simply exhibit more iterations.

These algorithms are easy to implement in existing simulation codes. The coding for the RDOF-NR method is independent of the linear equation solver. The RDOF-solver algorithm requires only a modification of the preconditioning phase of the solution. Of special note is the fact that neither of these algorithm requires any of the memory overhead of traditional adaptive implicit methods.

Acknowledgements

This research was supported by the US Department of Energy at the Los Alamos National Laboratory. The authors wish to thank Peter Lichtner and several anonymous reviewers for reviewing the paper and providing many helpful comments.

References