INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 7, 551–566 (1987)

APPLICATION OF CONJUGATE-GRADIENT-LIKE METHODS TO A HYPERBOLIC PROBLEM IN POROUS-MEDIA FLOW

UPUL R. B. OBEYSEKARE, MYRON B. ALLEN, RICHARD E. EWING AND JOHN H. GEORGE

University of Wyoming, Laramie, Wyoming 82071, U.S.A.

SUMMARY

This paper presents the application of a preconditioned conjugate-gradient-like method to a non-self-adjoint problem of interest in underground flow simulation. The method furnishes a reliable iterative solution scheme for the non-symmetric matrices arising at each iteration of the non-linear time-stepping scheme. The method employs a generalized conjugate residual scheme with nested factorization as a preconditioner. Model runs demonstrate significant computational savings over direct sparse matrix solvers.

KEY WORDS Generalized Conjugate Residual Method Nested Factorization Buckley–Leverett Equation Preconditioned Conjugate Gradients.

1. INTRODUCTION

Recent years have seen increasing interest in algorithms of the conjugate-gradient type for solving algebraic analogues of differential equations. This trend has been especially strong in petroleum reservoir simulation, where the need to solve large, sparse matrix equations arising from coupled sets of flow equations has sparked intensive research into iterative solution techniques. The use of the conjugate gradient method to solve linear systems $\mathbf{Ju} = -\mathbf{r}$ dates at least to Hestenes and Stiefel,[1] who examine the standard algorithms applicable to symmetric, positive-definite matrices. Such matrices arise in the approximate solution of many elliptic partial differential equations, Laplace's equation being a prototype. We owe to Reid[2] the view of the conjugate-gradient method as an iterative technique appropriate for sparse matrix systems and to Meijerink and van der Vorst[3] the development of a practical solver using preconditioning to speed convergence in systems involving large, symmetric, positive-definite matrices.

However, in fluid flow problems the differential operators are rarely self-adjoint, and as a result discrete approximations typically give rise to non-symmetric matrices. The importance of such applications has motivated the development of a variety of techniques, related to the conjugate-gradient method, that accommodate non-symmetric matrices. Among these are Manteuffel's Chebyshev iteration method,[4] Kershaw's application of the conjugate-gradient procedure to the normal equations $\mathbf{J}^{\mathrm{T}}\mathbf{Ju} = -\mathbf{J}^{\mathrm{T}}\mathbf{r}$,[5] Saad's incomplete orthogonalization methods,[6] the biconjugate gradient (BCG) method presented by Fletcher[7] and a class of preconditioned iterative methods based on Elman's generalized conjugate residual (GCR) method.[8,9] The GCR-based algorithms are attractive because none of them relies on computations involving $\mathbf{J}^{\mathrm{T}}\mathbf{J}$, whose condition number may be much larger than that of the non-symmetric matrix $\mathbf{J}$. Also, the GCR algorithm always converges provided the symmetric part of $\mathbf{J}$ is positive-definite, whereas with the BCG method

0271–2091/87/060551–16\$08.00

Received 16 December 1985
Revised 6 July 1986

no such guarantee exists.[10] As we shall review below, the use of an effective preconditioner is a key to efficient applications of this class of methods.

This paper presents an application of a GCR method, using nested factorization as a preconditioner, to the numerical solution of a highly non-symmetric problem associated with porous-media flows. We take as our model equation the non-linear hyperbolic conservation law:

$$\frac{\partial S}{\partial t}+\frac{\partial f(S)}{\partial x}+\frac{\partial f(S)}{\partial y}=q, \tag{1}$$

where f is a non-convex function of the principal unknown $S(\mathbf{x},t)$. This equation serves as a very simple analogue of the two-dimensional Buckley–Leverett problem for immiscible displacements in porous media.[11] Interest in conjugate-gradient-like methods for related problems has been growing in the petroleum reservoir simulation community; see for example References 12–16. To focus attention on the iterative linear solution technique, we have stripped away most of the detailed physics of the Buckley–Leverett problem while retaining the features of non-linearity and hyperbolicity that lead to numerical difficulties. The absence of second-order terms in equation (1) reflects one of the key assumptions behind the Buckley–Leverett model, namely that the effects of capillary pressure gradients in driving the flow are negligible compared with the effects of applied pressure gradients.[11] If we follow the analogy between equation (1) and an idealized waterflood in an oil reservoir, $S(\mathbf{x},t)$ signifies the water saturation, q represents injection and production rates and $f(S)$ is the fractional flow of water. Figure 1 depicts a typical fractional flow function. Equation (1) is a non-linear, first-order, hyperbolic partial differential equation written in

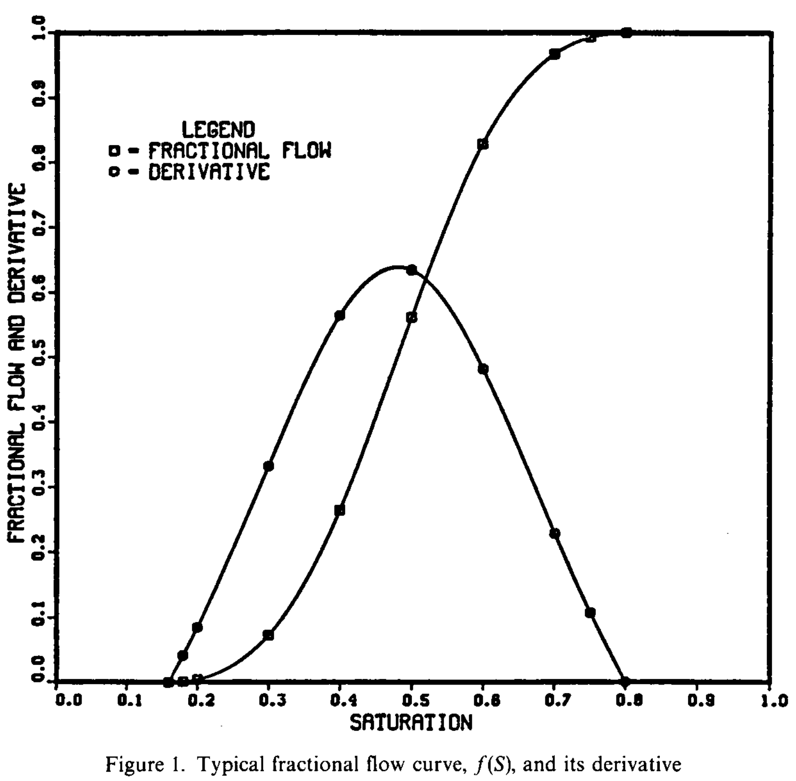

Figure 1. Typical fractional flow curve, $f(S)$, and its derivative

conservation form. Since the equivalent differential operator $\partial/\partial t + f'(S)(\partial/\partial x + \partial/\partial y)$ is not self-adjoint, discrete approximations to this equation will generally lead to non-symmetric matrix equations. Section 2 of this paper describes one class of discrete approximations using the method of finite differences.

Our discretization leads to a matrix equation $\mathbf{Ju} = -\mathbf{r}$ at each Newton-like iteration of each time step. As noted above, the hyperbolicity of equation (1) forces $\mathbf{J}$ to be non-symmetric. To solve the matrix equations efficiently, we use the GCR algorithm,[9] which works for non-symmetric matrices so long as their symmetric parts are positive definite. Section 3 describes the algorithm in detail and discusses some of its computational aspects.

One characteristic difficulty of conjugate-gradient-like methods is that they tend to converge quite slowly when the condition number of the matrix is large. This fact can cause trouble in practice, since in many problems the condition number increases as the spatial grid mesh becomes finer. The occurrence of poorly conditioned linear systems in discretized flow problems has prompted a great deal of interest in preconditioning methods to improve the convergence rates of iterative solution schemes. In section 4 we demonstrate the application of a preconditioner using nested factorization,[17] which is especially well suited to the block structures arising in finite-difference approximations.

Finally, in section 5, we discuss some sample problems solved using our approach. We illustrate, for example, the fact that the purely hyperbolic equation (1) exhibits uniqueness difficulties and that some device such as upstream weighting is needed to guarantee physically correct approximate solutions. We also examine the effects of capillary pressure gradients on the scheme, showing that when capillarity is significant upstream weighting is unnecessary. We conclude by comparing execution times for the preconditioned GCR technique with those obtained using a standard direct solution method. These comparisons demonstrate the efficacy of using nested factorization as a preconditioner in conjunction with GCR in hyperbolic or near-hyperbolic flow problems.

FINITE DIFFERENCE APPROXIMATION

Consider a rectangular spatial domain $\Omega \subset \mathbb{R}^2$ that can be written as the Cartesian product $(0, x_{\max}) \times (0, y_{\max})$ of two open intervals. Given uniform partitions $\mathscr{P}_x: 0 = x_0 < \cdots < x_i = i\Delta x < \cdots < x_I = x_{\max}$ and $\mathscr{P}_y: 0 = y_0 < \cdots < y_j = j\Delta y < \cdots < y_J = y_{\max}$, we can approximate spatial derivatives using various finite-difference analogues. Similarly, if we adopt a partition $\mathscr{P}_t: t_0 < \cdots < t_n = n\Delta t < \cdots$ of the time domain $T = (0, \infty)$, we can use a variety of difference expressions to approximate time derivatives:

$$\left.\frac{\partial S}{\partial t}\right|_{i,j}^{n+1} \simeq \frac{1}{\Delta t}(S_{i,j}^{n+1} - S_{i,j}^{n}),$$

$$\left.\frac{\partial f}{\partial x}\right|_{i,j}^{n+1} \simeq \frac{\theta}{\Delta x}(f_{i+1,j}^{n+1} - f_{i,j}^{n+1}) + \frac{1-\theta}{\Delta x}(f_{i,j}^{n+1} - f_{i-1,j}^{n+1}),$$

$$\left.\frac{\partial f}{\partial y}\right|_{i,j}^{n+1} \simeq \frac{\theta}{\Delta y}(f_{i,j+1}^{n+1} - f_{i,j}^{n+1}) + \frac{1-\theta}{\Delta y}(f_{i,j}^{n+1} - f_{i,j-1}^{n+1}).$$

In these equations $S_{i,j}^{n+1}$ denotes the approximate value of $S(i\Delta x, j\Delta y, n\Delta t)$ and $f_{i,j}^{n+1} \equiv f(S_{i,j}^{n+1})$. The parameter $\theta \in [0,1]$ indicates the weighting of the difference approximations in space. In particular, $\theta = 0, \frac{1}{2}, 1$ correspond to fully upstream, centred and fully downstream weighting, respectively, in cases where fluid velocity components in the x- and y-directions are positive.

This implicit scheme calls for the evaluation of the fractional flow function f at unknown arguments $S_{i,j}^{n+1}$. To accommodate this non-linearity, let us introduce an iterative technique for advancing from the known time level t_n to the unknown level t_{n+1}. We approximate the unknown value $S_{i,j}^{n+1}$ as follows:

$$S_{i,j}^{n+1} \simeq S_{i,j}^{n+1,m} + \delta S_{i,j}^{n+1,m+1},$$

where $S_{i,j}^{n+1,m}$ stands for the most recently known iterative value of $S_{i,j}^{n+1}$ and $\delta S_{i,j}^{n+1,m+1}$ represents an unknown iterative correction giving the new iterate $S_{i,j}^{n+1,m+1} = S_{i,j}^{n+1,m} + \delta S_{i,j}^{n+1,m+1}$. To start the iteration at each time step, we might, for example, set $S_{i,j}^{n+1,0} = S_{i,j}^{n}$. To obtain analogous expressions for the flux terms in equation (1) we can use the linear extrapolation

$$f_{i,j}^{n+1} \simeq f_{i,j}^{n+1,m} + f'(S_{i,j}^{n+1,m})\delta S_{i,j}^{n+1,m}$$

Here, as the notation suggests, $f_{i,j}^{n+1,m} \equiv f(S_{i,j}^{n+1,m})$ is the value of f as the known iterative stage.

Substituting these approximations into the partial differential equation (1) and rearranging yields the following difference formula:

$$\begin{aligned}&\left[-\left(\frac{1-\theta}{\Delta y}\right)f'(S_{i,j-1}^{n+1,m})\right]\delta S_{i,j-1}^{n+1,m+1} + \left[-\left(\frac{1-\theta}{\Delta x}\right)f'(S_{i-1,j}^{n+1,m})\right]\delta S_{i-1,j}^{n+1,m+1}\\ &+\left[\frac{1}{\Delta t}+\left(\frac{1-2\theta}{\Delta x}+\frac{1-2\theta}{\Delta y}\right)f'(S_{i,j}^{n+1,m})\right]\delta S_{i,j}^{n+1,m+1}\\ &+\left[\frac{\theta}{\Delta x}f'(S_{i+1,j}^{n+1,m})\right]\delta S_{i+1,j}^{n+1,m+1} + \left[\frac{\theta}{\Delta y}f'(S_{i,j+1}^{n+1,m})\right]\delta S_{i,j+1}^{n+1,m+1} = -r_{i,j}^{n+1,m}. \end{aligned} \tag{2}$$

Here

$$\begin{aligned} r_{i,j}^{n+1,m} \equiv &\frac{1}{\Delta t}(S_{i,j}^{n+1,m} - S_{i,j}^{n}) + \frac{\theta}{\Delta x}(f_{i+1,j}^{n+1,m} - f_{i,j}^{n+1,m})\\ &+\left(\frac{1-\theta}{\Delta x}\right)(f_{i,j}^{n+1,m} - f_{i-1,j}^{n+1,m}) + \frac{\theta}{\Delta y}(f_{i,j+1}^{n+1,m} - f_{i,j}^{n+1,m})\\ &+\left(\frac{1-\theta}{\Delta y}\right)(f_{i,j}^{n+1,m} - f_{i,j-1}^{n+1,m}) - q_{i,j}. \end{aligned}$$

Note that $r_{i,j}^{n+1,m}$ is the residual in the difference scheme at the mth or most recently known iterative stage. In practice we solve the system (2) of discrete equations iteratively for the corrections $\delta S_{i,j}^{n+1,m+1}$, stopping the iteration when $\max_{i,j}|r_{i,j}^{n+1,m+1}|$ falls below some prescribed tolerance.

Given appropriate boundary and initial conditions, the finite-difference scheme in equation (2) generates a system of linear equations having the form $\mathbf{J}^{n+1,m}\mathbf{u}^{n+1,m+1} = -\mathbf{r}^{n+1,m}$. Here $\mathbf{r}^{n+1,m}$ is the vector of residuals at the last known iteration, $\mathbf{J}^{n+1,m}$ is a pentadiagonal matrix of coefficients, and $\mathbf{u}^{n+1,m+1}$ represents the vector of unknown interative increments. This matrix equation resembles that arising in the standard Newton method for the iterative solution of non-linear algebraic systems, and thus $\mathbf{J}^{n+1,m}$ plays a role analogous to that of the Jacobian matrix. Equation (2) clearly shows that $\mathbf{J}^{n+1,m}$ will generally be non-symmetric, owing to the fact that the original differential operator in equation (1) is not self-adjoint.

3. GENERALIZED CONJUGATE RESIDUAL ALGORITHM

We have seen that the linearized iterative scheme arising from the finite-difference approximation of equation (1) requires the solution of a non-symmetric matrix equation having the form $\mathbf{Ju} = -\mathbf{r}$

at each iteration. Let us assume $\mathbf{u}, \mathbf{r} \in \mathbb{R}^d$ and $\mathbf{J} \in \mathbb{R}^{d \times d}$. The generalized conjugate residual (GCR) algorithm described by Elman[9] is essentially a generalization of the conjugate gradient algorithm to non-symmetric systems. The only requirement for GCR to be applicable is that the symmetric part symm $\mathbf{J} \equiv (\mathbf{J} + \mathbf{J}^T)/2$ be positive definite. As we shall discuss, this requirement allows an estimate of the convergence rate for GCR. Elman[9] presents numerical comparisons with other schemes, verifying the efficiency of the GCR method (although it is only fair to mention that in certain of these experiments Manteuffel's scheme[4] performed comparably well).

The idea behind GCR is to compute a sequence of approximate solutions $\mathbf{u}_{(0)}, \mathbf{u}_{(1)}, \ldots, \mathbf{u}_{(k)}, \ldots$ that minimize the Euclidean norm $\|\boldsymbol{\varepsilon}_{(k)}\|_2 = \|-\mathbf{r} - \mathbf{J}\mathbf{u}_{(k)}\|_2$ of the error over successively larger subsets of $\mathbb{R}^d$. Before making this idea precise, let us state the GCR algorithm.

Algorithm 1

Given a non-singular matrix $\mathbf{J} \in \mathbb{R}^{d \times d}$, $\mathbf{r} \in \mathbb{R}^d$, and an initial guess $\mathbf{u}_{(0)} \in \mathbb{R}^d$, with symm $\mathbf{J}$ positive definite, the following algorithm implements the GCR method for solving the system $\mathbf{Ju} = -\mathbf{r}$ to within a given tolerance $\tau > 0$:

```
$k := 0$,
$\varepsilon_{(0)} := -\mathbf{r} - \mathbf{J}\mathbf{u}_{(0)}$   (initial error),
$\mathbf{q}_{(0)} := \varepsilon_{(0)}$   (initial search direction),
$\mathbf{v}_{(0)} := \mathbf{J}\mathbf{q}_{(0)}$,
Do while $\|\varepsilon_{(k)}\|_2 \geqslant \tau$
  $\alpha_{(k)} := \varepsilon_{(k)}^T \mathbf{v}_{(k)} / (\mathbf{v}_{(k)}^T \mathbf{v}_{(k)})$,
  $\mathbf{u}_{(k+1)} := \mathbf{u}_{(k)} + \alpha_{(k)} \mathbf{q}_{(k)}$   (New approximate solution),
  $\varepsilon_{(k+1)} := \varepsilon_{(k)} - \alpha_{(k)} \mathbf{v}_{(k)}$   (New error),
  for $l = 0, \ldots, k$
    $\beta_{(l,k+1)} := (\mathbf{J}\varepsilon_{(k+1)})^T \mathbf{v}_l / (\mathbf{v}_l^T \mathbf{v}_l)$,
  $\mathbf{q}_{(k+1)} := \varepsilon_{(k+1)} := \sum_{l=0} \beta_{(l,k+1)} \mathbf{q}_{(l)}$   (New search direction),
  $\mathbf{v}_{(k+1)} := \mathbf{J}\varepsilon_{(k+1)} - \sum_{l=0}^{k} \beta_{(l,k+1)} \mathbf{v}_{(l)}$,
  $k := k + 1$.
```

Each stage k of this scheme determines a search direction vector $\mathbf{q}_{(k)}$ that is $\mathbf{J}$-orthogonal to each of the previous search directions $\mathbf{q}_{(l)}, 0 \leqslant l \leqslant k-1$, meaning that $(\mathbf{J}\mathbf{q}_{(k)})^T \mathbf{J}\mathbf{q}_{(l)} = 0$ for $k \neq l$. The choice of the scalar $\alpha_{(k)}$ guarantees that the new iterate $\mathbf{u}_{(k+1)}$ will minimize $\|\varepsilon_{(k+1)}\|_2$ over the affine subspace $\mathbf{u}_{(0)} + \text{span}\{\mathbf{q}_{(0)}, \ldots, \mathbf{q}_{(k)}\}$. Theoretically, then, the algorithm must terminate in d stages, since it will have minimized the error over all of $\mathbb{R}^d$. In practice, however, the finite precision of machine arithmetic causes a loss of strict orthogonality in the search directions, so GCR behaves computationally like a true iterative scheme.

When we view GCR as an iterative scheme rather than as a procedure terminating after finitely many stages, it becomes important to guarantee that the scheme converges rapidly. There are two reasons for this. The first stems from the observation that Algorithm 1 requires storage of the search direction $\mathbf{q}_{(k)}$ for each iteration k. This requirement does not arise in standard conjugate-gradient methods for symmetric matrices $\mathbf{J}$. However, it is common in related schemes for non-symmetric matrices,[4,6,7] the main exceptions being methods based on calculations with $\mathbf{J}^T\mathbf{J}$. The extra storage required by truly non-symmetric schemes can be computationally disastrous if the system to be solved takes many iterations to achieve numerical convergence. However, if we can speed convergence so that the method requires only a few iterations per time step, then the extra storage needed will amount to a tolerable increment in computational overhead.

The second reason for wanting rapid convergence is that we are interested in solving large sets of linear equations. Thus the dimension d of the system is generally large, and an $O(d)$ iteration count is unacceptable. This observation is especially relevant in the numerical solution of flow equations, since accurate solutions usually demand fine spatial grids and hence large-order matrices. Elman[9] gives an estimate of the convergence rate for GCR, assuming that symm $\mathbf{J}$ is positive definite. Let us call skew $\mathbf{J} \equiv -\frac{1}{2}(\mathbf{J}-\mathbf{J}^T)$. Also, for any matrix $\mathbf{N}\in\mathbb{R}^{d\times d}$, denote by $\{\lambda_q(\mathbf{N})\}_{q=1}^{d}$ the eigenvalues of $\mathbf{N}$; let $\lambda_{\max}(\mathbf{N})$ and $\lambda_{\min}(\mathbf{N})$ be the eigenvalues of $\mathbf{N}$ having the largest and smallest absolute values, respectively, and call $\rho(\mathbf{N}) = |\lambda_{\max}|$ the spectral of $\mathbf{N}$. Then the error at the kth iterative stage obeys the following bound:

$$\|\boldsymbol{\varepsilon}_{(k)}\|_2 \leqslant \left\{1-\frac{\lambda_{\min}^2(\text{symm }\mathbf{J})}{\lambda_{\min}(\text{symm }\mathbf{J})\lambda_{\max}(\text{symm }\mathbf{J})+[\rho(\text{skew }\mathbf{J})]^2}\right\}^{k/2}\|\varepsilon_{(0)}\|_2.$$

Obviously, we would like the expression in braces to be as small as possible for rapid convergence. There are two important ways in which this estimate can indicate slow convergence. The first is when $\rho(\text{skew }\mathbf{J}) \gg \lambda_{\min}(\text{symm }\mathbf{J})$, which occurs when the matrix $\mathbf{J}$ is strongly non-symmetric. The second is when $\lambda_{\max}(\text{symm }\mathbf{J})/\lambda_{\min}(\text{symm }\mathbf{J}) \gg 1$, that is when symm $\mathbf{J}$ has a large condition number. Thus, roughly speaking, we can expect GCR to converge slowly when the matrix equation to be solved is poorly conditioned. Typical discrete methods for spatial differential operators yield matrices whose condition numbers grow in inverse proportion to the discretization error, so that fine grids imply large condition numbers. Therefore we can expect to encounter slower convergence rates for GCR precisely when we seek more accurate solutions of flow equations. This fact motivates the use of preconditioning methods to reduce poorly conditioned matrix equations $\mathbf{Ju} = -\mathbf{r}$ to equivalent systems with better condition numbers.

4. PRECONDITIONING BY NESTED FACTORIZATION

Our basic strategy in preconditioning is to replace a possibly ill-conditioned system $\mathbf{Ju} = -\mathbf{r}$ by an equivalent system $\hat{\mathbf{J}}^{-1}\mathbf{Ju} = -\hat{\mathbf{J}}^{-1}\mathbf{r}$. For this strategy to be successful computationally, the preconditioner $\hat{\mathbf{J}}^{-1}$ must be chosen to satisfy two criteria. First, the matrix $\hat{\mathbf{J}}^{-1}\mathbf{J}$ must have a relatively small condition number; secondly, $-\hat{\mathbf{J}}^{-1}\mathbf{v}$ should be easy to compute for arbitrary $\mathbf{v}\in\mathbb{R}^d$. In a heuristic sense, $\hat{\mathbf{J}}^{-1}$ should be 'close to' the actual inverse of $\mathbf{J}$.

Nested factorization is an algorithm introduced by Appleyard *et al.*[17] that takes advantage of the nested block structure of finite-difference matrices to compute $\hat{\mathbf{J}}^{-1}$. We shall review the algorithm allowing three space dimensions for generality. Consider the $m \times n \times p$ finite-difference grid shown in Figure 2. Such a grid, together with a seven-point difference scheme, yields a matrix having the heptadiagonal structure depicted in Figure 3. This matrix exhibits a nested tridiagonal form in the following sense: the small blocks $\mathbf{G}_j\in\mathbb{R}^{m\times m}$ along the diagonal are tridiagonal with non-zero l_i, d_i, and u_i in the ith row; the large blocks $\mathbf{A}_k\in\mathbb{R}^{nm\times nm}$ along the diagonal are block tridiagonal with non-zero matrices $\mathbf{M}_j, \mathbf{G}_j, \mathbf{V}_j\in\mathbb{R}^{nm\times m}$ in the jth row of small blocks, and the entire matrix $\mathbf{J}\in\mathbb{R}^{pnm\times pnm} = \mathbb{R}^{d\times d}$ is block-tridiagonal with non-zero matrices $\mathbf{Q}_k, \mathbf{A}_k, \mathbf{W}_k\in\mathbb{R}^{nm\times nm}$ in the kth row of large blocks. For convenience, let us denote the $m\times m$ matrices containing the subdiagonal entries l_i and superdiagonal entries u_i of the small blocks $\mathbf{G}_j$ by $\mathbf{L}_j$ and $\mathbf{U}_j$, respectively.

The observation underlying nested factorization is that LDU decompositions of tridiagonal structures are very easy to compute and invert. Thus we can compute the LDU factorizations of the small blocks $\mathbf{G}_j$, use these to compute 'approximate' LDU factorizations of the large blocks $\mathbf{A}_k$, then use these to compute an approximate LDU factorization of the entire matrix $\mathbf{J}$. At each level in this nested structure the algorithm formally uses computations that mimic the Cholesky

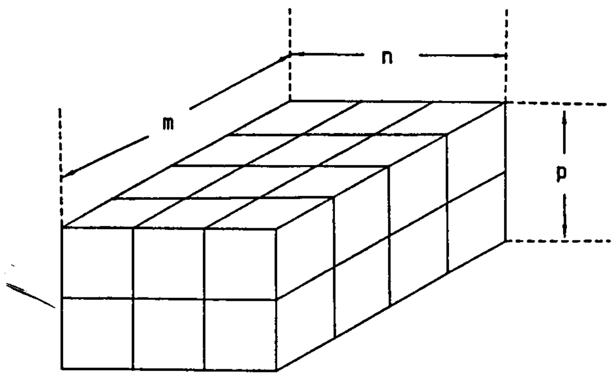

Figure 2. Typical three-dimensional finite-difference grid

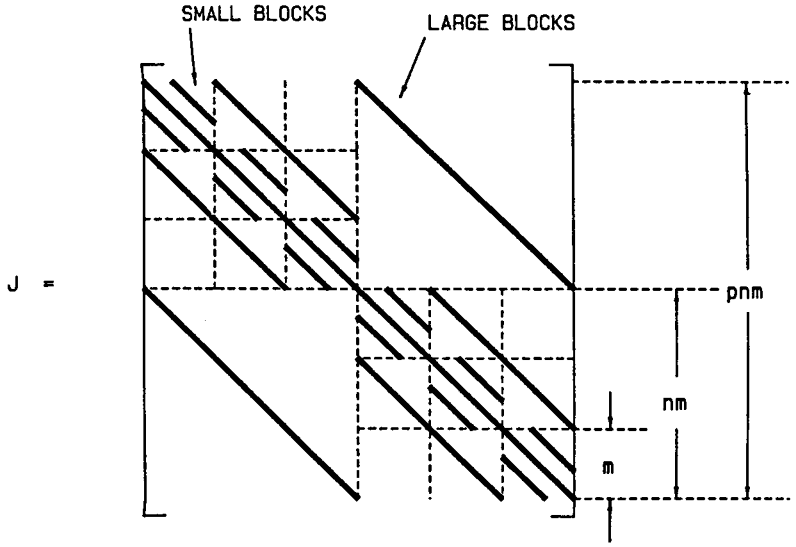

Figure 3. Block structure of the iteration matrix **J** associated with the grid drawn in Figure 2

decomposition of a tridiagonal matrix. There is one modification, however. As Appleyard *et al.*[17] explain, to preserve the mass-conserving properties of the original problem $\mathbf{Ju} = -\mathbf{r}$ at each iteration, one should construct a preconditioner $\hat{\mathbf{J}}$ whose column sums equal those of $\mathbf{J}$. This requirement leads to a modification of the Cholesky decompositions for the small blocks $\mathbf{G}_j$. The precise statement of the nested factorization algorithm is as follows.

Algorithm 2

Given a heptadiagonal matrix $\mathbf{J} \in \mathbb{R}^{pnm \times pnm}$ having the nested block structure described above, the following algorithm produces an approximate LDU factorization $\hat{\mathbf{J}}$ of $\mathbf{J}$ using nested factorization:

```
For $k = 1, \ldots, p$  (loop over large blocks)
  For $j = 1, \ldots, n$  (loop over small blocks)
    For $i = 1, \ldots, m$  (loop over rows in each small block)
      $\hat{\imath} := (k-1)n + i$  (global row number),
```

$$b_i := d_i - \frac{l_i u_{i-1}}{\gamma_{i-1}} - \sum_{\text{col}nm+i} (\mathbf{M}_j \hat{\mathbf{G}}_{j-1}^{-1} \mathbf{V}_{j-1}) - \sum_{\text{col}nm+i} (\mathbf{N}_k \hat{\mathbf{A}}_{k-1}^{-1} \mathbf{W}_{k-1}),$$

$\mathbf{B}_j := \text{diag}(b_1, \ldots, b_m) \in \mathbb{R}^{m \times m}$,

$\hat{\mathbf{G}}_j := (\mathbf{L}_j + \mathbf{B}_j)\mathbf{B}_j^{-1}(\mathbf{B}_j + \mathbf{U}_j) \in \mathbb{R}^{m \times m}$ (LDU factorization of $\mathbf{G}_j$ modified by column sum criterion)

$\mathbf{G} := \text{diag}(\hat{\mathbf{G}}_1, \ldots, \hat{\mathbf{G}}_n) \in \mathbb{R}^{nm \times nm}$,

$\mathbf{M} := \text{subdiag}(\mathbf{M}_2, \ldots, \mathbf{M}_n) \in \mathbb{R}^{nm \times nm}$,

$\mathbf{V} := \text{superdiag}(\mathbf{V}_1, \ldots, \mathbf{V}_{n-1}) \in \mathbb{R}^{nm \times nm}$,

$\hat{\mathbf{A}}_k := (\mathbf{M} + \mathbf{G})\mathbf{G}^{-1}(\mathbf{G} + \mathbf{V}) \in \mathbb{R}^{nm \times nm}$ (approximate LDU factorization of large block $\mathbf{A}_k$),

$\mathbf{A} := \text{diag}(\hat{\mathbf{A}}_1, \ldots, \hat{\mathbf{A}}_p) \in \mathbb{R}^{pnm \times pnm}$,

$\mathbf{Q} := \text{subdiag}(\mathbf{Q}_2, \ldots, \mathbf{Q}_p) \in \mathbb{R}^{pnm \times pnm}$,

$\mathbf{W} := \text{superdiag}(\mathbf{W}_1, \ldots, \mathbf{W}_{p-1}) \in \mathbb{R}^{pnm \times pnm}$,

$\hat{\mathbf{J}} := (\mathbf{Q} + \mathbf{A})\mathbf{A}^{-1}(\mathbf{A} + \mathbf{W}) \in \mathbb{R}^{pnm \times pnm} = \mathbb{R}^{d \times d}$ (approximate LDU factorization of $\mathbf{J}$).

(In cases where an index variable such as $j-1$ or $k-1$ refers to an undefined index value, the indexed variable is understood to be zero. The notation $\Sigma_{\text{col}e}\mathbf{N}$ signifies the sum of the entries in the eth column of the matrix $\mathbf{N}$.)

Once we have computed the preconditioner $\hat{\mathbf{J}}$, we must incorporate it into the GCR solution of the system $\hat{\mathbf{J}}^{-1}\mathbf{J}\mathbf{u} = -\hat{\mathbf{J}}^{-1}\mathbf{r}$. Because the nested factorization algorithm produces $\hat{\mathbf{J}}$ in LDU form, the computation of $\hat{\mathbf{J}}^{-1}\mathbf{v}$ is quite efficient for any $\mathbf{v} \in \mathbb{R}^d$. The resulting preconditioned GCR algorithm is as follows.

Algorithm 3

Given the non-singular matrix $\mathbf{J} \in \mathbb{R}^{d \times d}$, $\mathbf{r} \in \mathbb{R}^d$, and $\mathbf{u}_{(0)} \in \mathbb{R}^d$ as in Algorithm 1, together with a preconditioner $\hat{\mathbf{J}} \in \mathbb{R}^{d \times d}$, the following algorithm implements the preconditioned GCR method for solving $\mathbf{J}\mathbf{u} = -\mathbf{r}$ iteratively to within a tolerance $\tau > 0$:

```
$\kappa := 0$,
$\boldsymbol{\varepsilon}_{(0)} := \hat{\mathbf{J}}^{-1}(-\mathbf{r} - \mathbf{J}\mathbf{u}_{(0)})$,
$\mathbf{q}_{(0)} := \boldsymbol{\varepsilon}_{(0)}$,
$\mathbf{v}_{(0)} := \hat{\mathbf{J}}^{-1}\mathbf{J}\mathbf{q}_{(0)}$,
Do while $\|\boldsymbol{\varepsilon}_{(i)}\|_2 \geqslant \tau$
    $\alpha_{(k)} := \boldsymbol{\varepsilon}_{(k)}^{\mathrm{T}}\mathbf{v}_{(k)}/(\mathbf{v}_{(k)}^{\mathrm{T}}\mathbf{v}_{(k)})$,
    $\mathbf{u}_{(k+1)} := \mathbf{u}_{(k)} + \alpha_{(k)}\mathbf{q}_{(k)}$,
    $\boldsymbol{\varepsilon}_{(k+1)} := \boldsymbol{\varepsilon}_{(k)} - \alpha_k\mathbf{v}_{(k)}$,
    for $l = 0, \ldots, k$
        $\beta_{(lk+1)} := (\hat{\mathbf{J}}^{-1}\mathbf{J}\boldsymbol{\varepsilon}_{(k+1)})^{\mathrm{T}}\mathbf{v}_{(l)}/(\mathbf{v}_{(l)}^{\mathrm{T}}\mathbf{v}_{(l)})$,
    $\mathbf{q}_{(k+1)} := \boldsymbol{\varepsilon}_{(k+1)} - \sum_{l=0}^{k} \beta_{(l,k+1)}\mathbf{q}_{(l)}$,
    $\mathbf{v}_{(k+1)} := \hat{\mathbf{J}}^{-1}\mathbf{J}\boldsymbol{\varepsilon}_{(k+1)} - \sum_{l=0}^{k} \beta_{(l,k+1)}\mathbf{v}_{(l)}$,
    $k := k + 1$.
```

5. NUMERICAL RESULTS[18]

We have examined numerical results for the methods described above using an initial-boundary-value problem roughly representative of a quarter-five-spot pattern displacement in an oilfield. Specifically, we pose equation (1) on the unit square $\Omega = (0,1) \times (0,1)$ on the time interval $T = (0, \infty)$ with no-flow boundary conditions,

$$\frac{\partial S}{\partial x}(0,y) = \frac{\partial S}{\partial x}(1,y) = \frac{\partial S}{\partial y}(x,0) = \frac{\partial S}{\partial y}(x,1) = 0,$$

and the initial conditions $S(x,y,0) = S_{\mathrm{wr}}, (x,y) \in \Omega$. S_{wr} stands for the minimum value of water saturation. For the fractional flow function $f(S)$ we use the model curve[19]

$$f(S) = \frac{(S - S_{\mathrm{wr}})^2}{(S - S_{\mathrm{wr}})^2 + (1 - S_{\mathrm{or}} - S)^2},$$

where $S_{\mathrm{wr}} = 0{\cdot}16$ and $S_{\mathrm{or}} = 0{\cdot}20$. For the source–sink term we assume constant injection and production rates at the wells (0, 0) and (1, 1):

$$q(x,y) = \delta(0,0) - \delta(1,1),$$

where δ stands for the Dirac distribution. In a finite-difference discretization with a rectangular grid whose nodes are ordered lexicographically, this source–sink term has the algebraic analogue $(1, 0, \ldots, 1)^{\mathrm{T}}$.

Figures 4–6 show saturation profiles at various time levels for this problem, computed on a 32×32 node spatial grid with $\Delta t = 0{\cdot}01$. In this case $\theta = 0{\cdot}3$, giving a slightly upstream-weighted scheme in keeping with the conventional approach in oil-reservoir simulation. The saturation

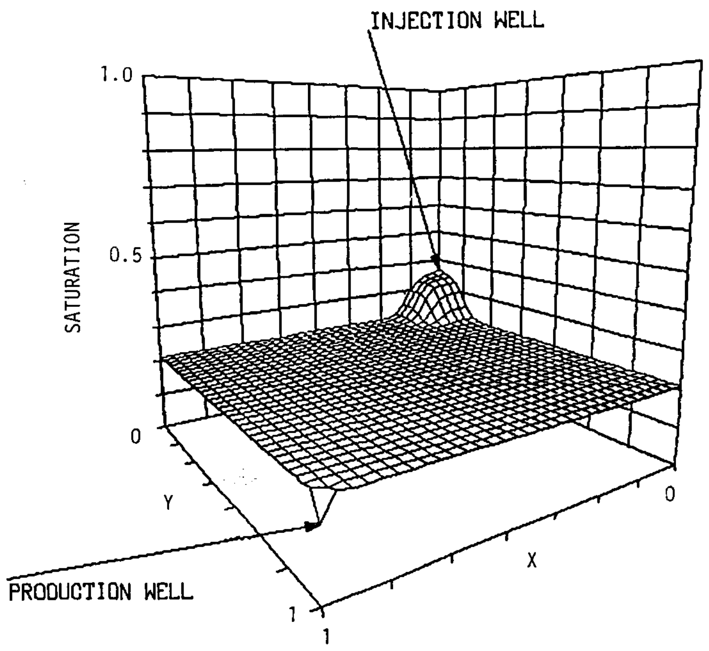

Figure 4. Numerical solution to the model problem on a 32×32 grid after 30 time steps ($\Delta t = 0{\cdot}01$), computed using an upstream weighted approximation ($\theta = 0{\cdot}3$)

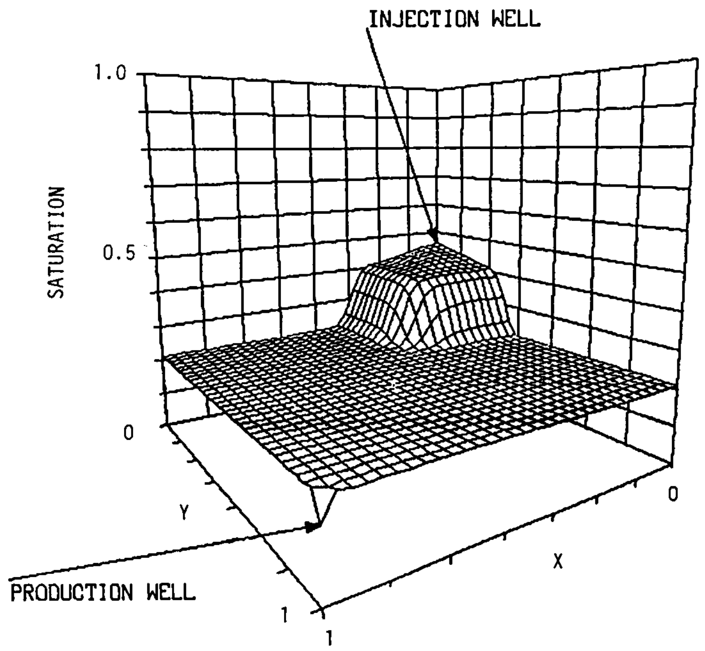

Figure 5. Numerical solution to the model problem after 50 time steps ($\Delta t = 0{\cdot}01$), computed using $\theta = 0{\cdot}3$

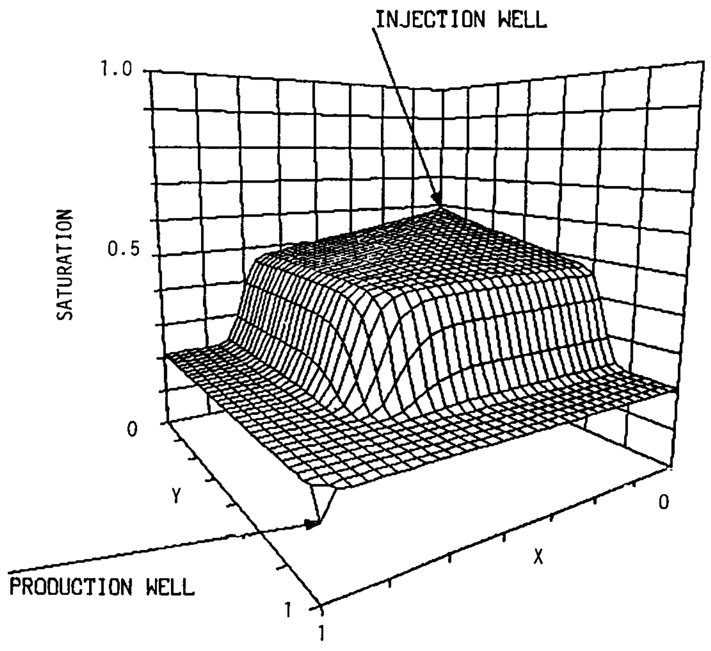

Figure 6. Numerical solution to the model problem after 75 time steps ($\Delta t = 0{\cdot}01$), computed using $\theta = 0{\cdot}3$

wave shown here is physically reasonable for a displacement having a favourable mobility ratio, with the exception of two types of error that are typical in finite-difference formulations of this problem. First, the use of upstream weighting induces numerical dissipation having magnitude $O(\Delta x + \Delta y)$ as discussed by Lantz.[20] Thus the saturation front, which should be a shock, appears smeared over several grid cells. The diagonal profiles in Figure 7 illustrate this smearing. Secondly, the saturation wave exhibits some grid-orientation bias: the saturation front tends to align itself with the grid-cell boundaries, so that we can expect a grid with a different orientation to produce somewhat different results. Although this effect may not be critical in simple problems such as this one, it can lead to serious distortions of physics in more complicated flows with chemical reactions and interphase mass transfer. It is only fair to note, however, that these difficulties afflict most upstream-weighted finite-difference discretizations of oilfield flows and are independent of the preconditioned GCR method.

Figures 8–10 show saturation profiles generated at various time levels using the same data as for Figures 4–6, with the exception that $\theta = 0{\cdot}5$. This centred difference scheme, although offering $O(\Delta x^2 + \Delta y^2)$ truncation error, leads to convergence difficulties with hyperbolic conservation laws such as the Buckley–Leverett equation. The oscillatory behaviour apparent in Figure 6 is evidence of these difficulties and occurs whether we use the preconditioned GCR method or, for example, a direct band matrix solver. Indeed, convergence failures in hyperbolic or near-hyperbolic oilfield flows have led petroleum reservoir engineers to adopt upstream weighting almost universally. It is worth mentioning that his circumstance has motivated a substantial body of research into the development of upstream-weighted numerical schemes that mitigate the

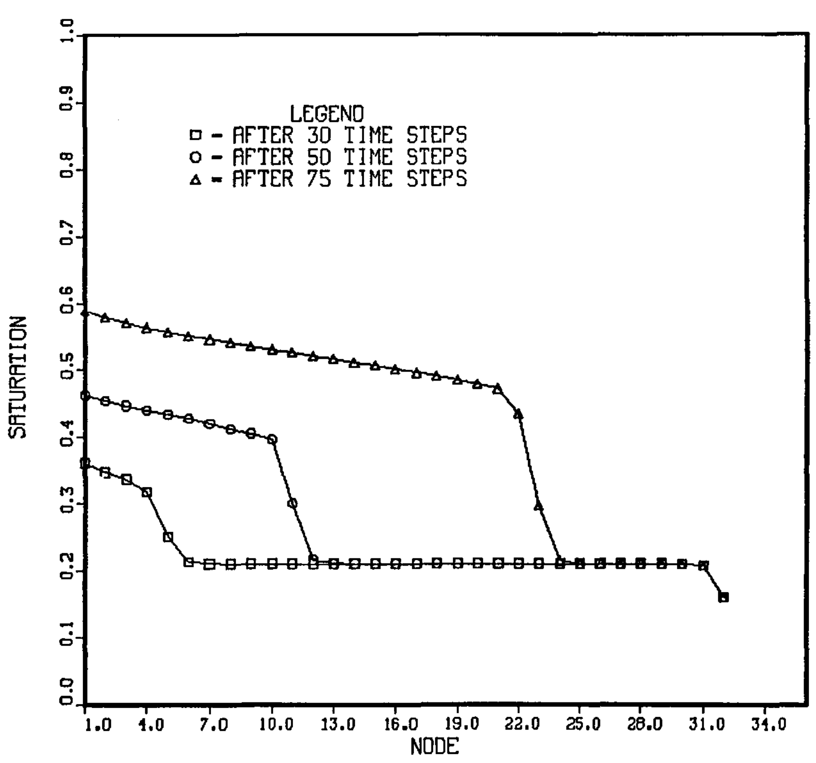

Figure 7. Profile of water saturations along the diagonal ($x = y$) at various time levels, computed using $\theta = 0$

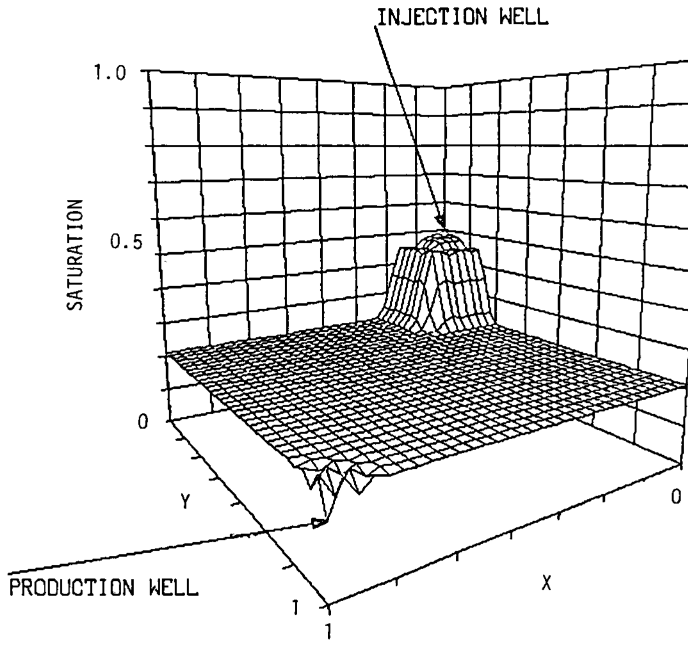

Figure 8. Numerical solution to the model problem after time steps, using the same data as in Figure 4 except that $\theta = 0{\cdot}5$

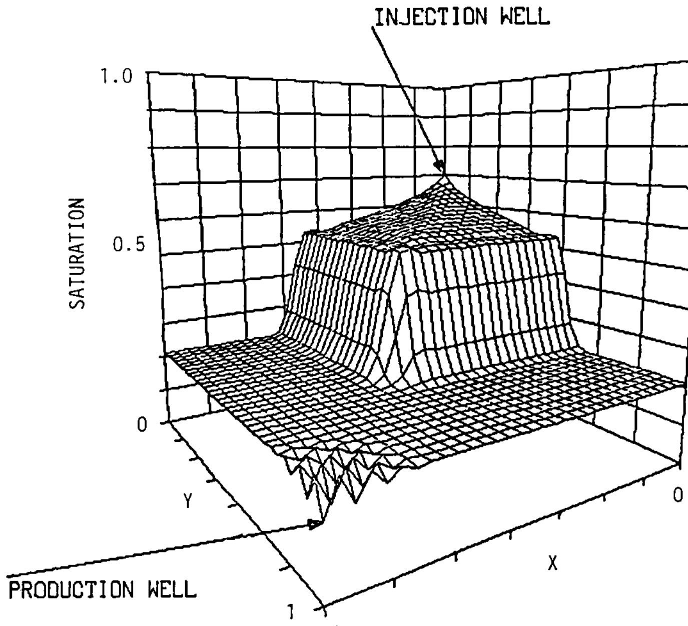

Figure 9. Numerical solution to the model problem after 50 time steps, using the same data as in Figure 5 except that $\theta = 0{\cdot}5$

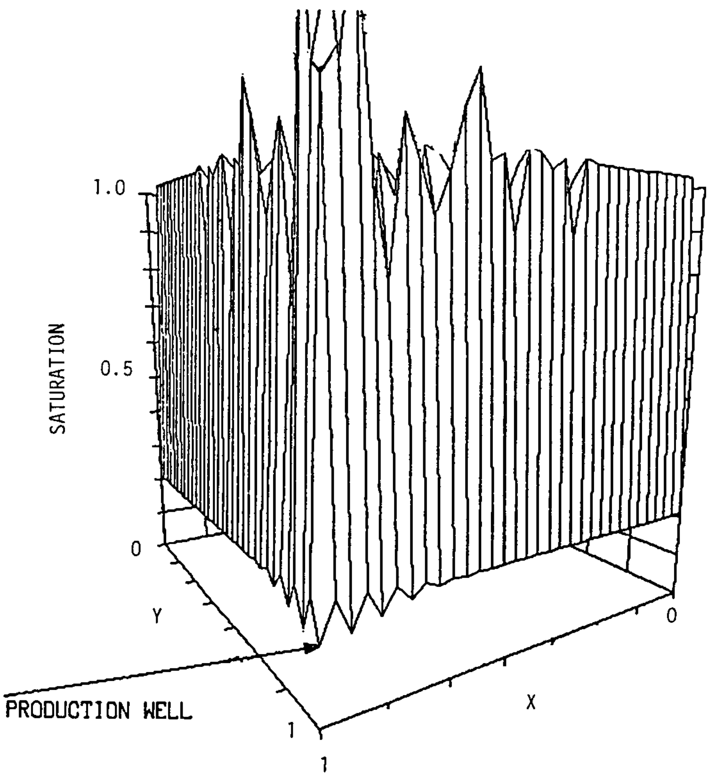

Figure 10. Numerical solution to the model problem after 75 time steps, using the same data as in Figure 6 except that $\theta = 0{\cdot}5$

numerical diffusion and grid-orientation effects evident in Figures 4–7; see Reference 21 for a review.

We have also briefly investigated the effect of capillarity on the numerical solution. In the capillarity-free case, upstream weighting ensures convergence through the addition of an artificial dissipative term proportional to the grid spacing.[22] In theory, if capillary pressure gradients are sufficiently large, the physical dissipation will stabilize the numerics. In this case, finite-difference schemes for the Buckley–Leverett problem will converge without upstream weighting. The following extension of equation (1) incorporates a simple model of capillarity:

$$\frac{\partial S}{\partial t}+\frac{\partial f}{\partial x}+\frac{\partial f}{\partial y}-C\left(\frac{\partial^2 S}{\partial x^2}+\frac{\partial^2 S}{\partial y^2}\right)=q,$$

where $C > 0$. To approximate the additional term, we use standard central differences, which have a truncation error that is $O(\Delta x^2 + \Delta y^2)$. For example

$$\left.\frac{\partial^2 S}{\partial x^2}\right|_{i,j}^{n+1} \simeq \frac{S_{i-1,j}^{n+1}-S_{i,j}^{n+1}+S_{i+1,j}^{n+1}}{\Delta x^2},$$

and similarly for $\partial^2 S/\partial y^2$. Figures 11 and 12 show a perspective plot and diagonal profiles, respectively, of the approximate water saturation computed using this scheme on a 32×32 grid with $C = 0{\cdot}005$ and $\theta = 0{\cdot}5$. Notice that centred differences give reasonable results in this case, in

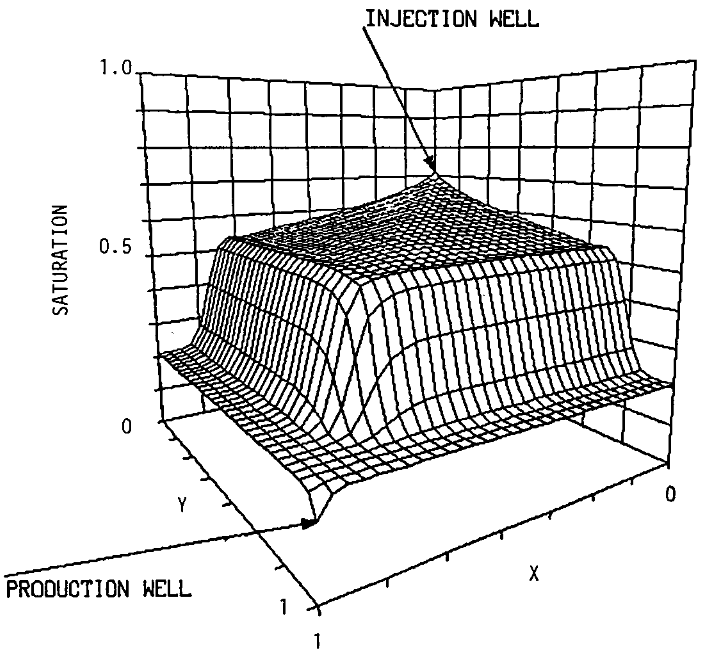

Figure 11. Numerical solution to the model problem after 75 time steps, using the same data as in Figure 10 except for the addition of a capillarity parameter $C = 0{\cdot}005$

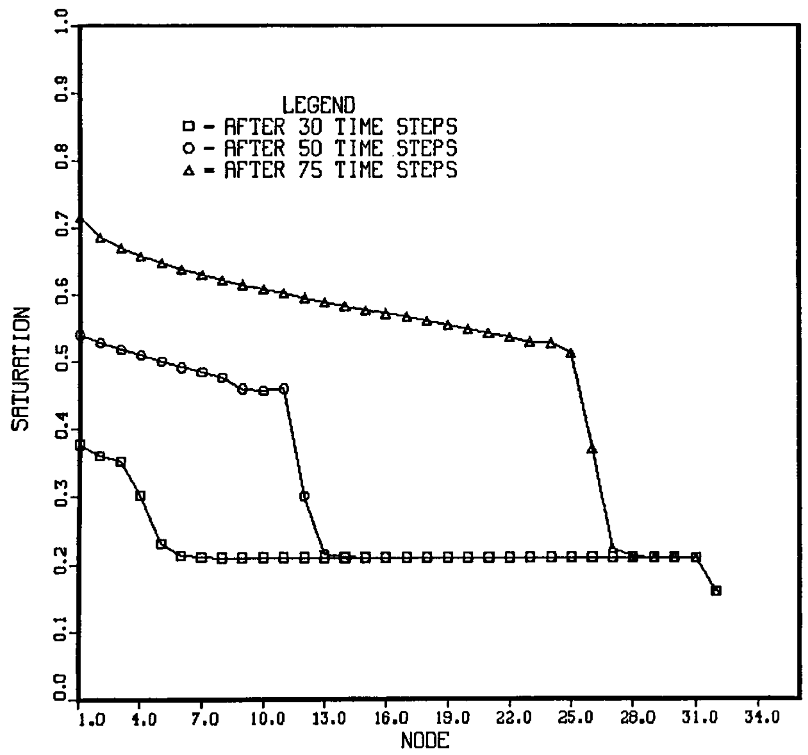

Figure 12. Profile of water saturations along the diagonal ($x = y$) at various time levels, computed using $\theta = 0{\cdot}5$ and $C = 0{\cdot}005$

Table I. Comparison of run times for the model problem using the preconditioned GCR algorithm and direct solvers

	$\theta = 0{\cdot}5$			$\theta = 0{\cdot}3$		
	Time steps			Time steps		
Method	30	50	75	30	50	75
PCG (7 Bands)	1·816	3·607	5·916	1·659	2·785	5·047
IMSL LEQT28 (7 Bands)	10·449	20·13	32·659	11·611	20·176	31·041
PCG (5 Bands)	1·357	2·633	4·427	1·231	2·094	3·736
IMSL LEQT18 (5 Bands)	1·737	3·238	5·192	1·787	3·117	4·784
IMSL LEQT28 (5 Bands)	2·468	4·796	7·813	2·790	4·845	7·600

Time in CP seconds

contrast to the highly oscillatory results generated by central differences for the capillarity-free case.

The preconditioned GCR method offers significant computational savings over conventional direct solution methods in the inversion of the finite-difference matrices at each Newton-like iteration of this problem. We compared CPU times required on a Cyber 760 for the solution of the problem discussed above, using square grid having eight nodes on a side, with the preconditioned GCR method and the IMSL asymmetric band matrix solvers LEQT1B and LEQT2B. Because of the relatively poor conditioning at each iteration, the direct solver required iterative improvement, and hence only the results obtained using LEQT2B should be considered valid. Note also that the IMSL subroutines are not the most efficient direct algorithms available for our matrix structure. However, they have the virtue of being widely available and thus provide useful bench-mark calculations.

Table I displays the results for several time levels in the calculation. As the data indicate, for the 8×8 model problem the preconditioned GCR algorithm ('PGC') requires roughly half the time needed by the direct solver when each subroutine receives pentadiagonal or five-banded matrices. To estimate the corresponding run times for a three-dimensional problem, we also sent each solver heptadiagonal or seven-banded matrices. In this case the preconditioned GCR code cut CPU time by a factor of six. On grids larger than 8×8 we can expect even greater savings.

6. CONCLUSIONS

The results reported here demonstrate the efficacy of conjugate-gradient-like methods in the solution of matrix equations arising from flow equations involving non-self-adjoint operators. The preconditioned GCR method presented above gives fast, accurate solutions to discrete analogue of a simple hyperbolic problem idealizing the Buckley–Leverett equations governing two-phase oil reservoir flow. The particular discretizations used in this work pose a rather strenuous test, since the simple upstream weighting scheme employed imposes strong directional tendencies in the numerics. Based on the results, we can expect the preconditioned GCR method to be effective in more complex and veracious models of underground flow.

ACKNOWLEDGEMENTS

The U.S. National Science Foundation supported this work through grant number DMS-8504360. The U.S. Air Force Office of Scientific Research provided support through grant number AFOSR-85-0117. The U.S. Army Research Office provided support through grant number DAAG29-84-K-0002.

REFERENCES

1. M. R. Hestenes and E. Stiefel, 'Methods of conjugate gradients for solving linear systems', *NBS J. Res.*, **49**, 409–436 (1952).
2. J. K. Reid, 'On the method of conjugate gradients for the solution of large sparse systems of linear equations', in J. K. Reid (ed), *Large Sparse Sets of Linear Equations*, Academic Press, New York, 1971, 231–254.
3. J. A. Meijerink and H. A. van der Vorst, 'An iterative solution method for linear systems in which the coefficient matrix is a symmetric *M*-matrix' *Math. Comp.*, **31**, 148–162 (1977).
4. T. A. Manteuffel, 'The Tchebychev iteration method for nonsymmetric linear systems', *Numer. Math.*, **28**, 307–327 (1977).
5. D. S. Kershaw, 'The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations', *J. Comp. Phys.*, **26**, 43–65 (1978).
6. Y. Saad, 'Practical use of some Krylov subspace methods for solving indefinite and nonsymmetric linear systems', *SIAM J. Sci. Stat. Comp.*, **5**, 203–228 (1984).
7. R. Fletcher, 'Conjugate gradient methods for indefinite systems', in G. A. Watson (ed.), *Numerical Analysis, Dundee 1975*, Springer-Verlag, New York, 1976, pp. 73–89.
8. S. C. Eisenstat, H. C. Elman and M. H. Schultz, 'Variational iterative methods for nonsymmetric systems of linear equations', *SIAM J. Numer. Anal.*, **20**, 345–357 (1983).
9. H. C. Elman, 'Preconditioned conjugate-gradient methods for nonsymmetric systems of linear equations', *Research Report 203*, Yale University Department of Computer Science, New Haven, Connecticut, 1981.
10. H. C. Elman, 'Iterative methods for non-self-adjoint elliptic problems', in G. Birkhoff and A. Schoenstadt (eds), *Elliptic Problem Solvers II*, Academic Press, Orlando, Florida, 1984, pp. 271–284.
11. S. E. Buckley and M. C. Leverett, 'Mechanism of fluid displacement in sands', *Trans. A.I.M.E.*, **146**, 107–116 (1941).
12. A. Behie, 'Comparison of nested factorization, constrained pressure residual, and incomplete factorization preconditioning', in L. C. Young *et al.* (eds), *Proceedings of the Eighth SPE Symposium on Reservoir Simulation*, Society of Petroleum Engineers, Dallas, 1985, pp. 355–374.
13. E. J. Northrup and P. T. Wood, 'Application of preconditioned conjugate-gradient-like methods in reservoir simulation', in L. C. Young *et al.* (eds), *Proceedings of the Eighth SPE Symposium on Reservoir Simulation*, Society of petroleum Engineers, Dallas, 1985, pp. 375–386.
14. H. D. Simon, 'Incomplete LU preconditioners for conjugate-gradient-type iterative methods', in L. C. Young *et al.* (eds), *Proceedings of the Eighth SPE Symposium on Reservoir Simulation*, Society of Petroleum Engineers, Dallas, 1985, pp. 387–396.
15. S. C. Eisenstat, H. C. Elman and M. H. Schultz, 'Block-preconditioned conjugate gradient-like methods for numerical reservoir simulation', in L. C. Young *et al.* (eds), *Proceedings of the Eighth SPE Symposium on Reservoir Simulation*, Society of Petroleum Engineers, Dallas, 1985, pp. 397–406.
16. J. R. Wallis, R. P. Kendall and T. E. Little, 'Constrained residual acceleration of conjugate residual methods', in L. C. Young *et al.* (eds), *Proceedings of the Eighth SPE Symposium on Reservoir Simulation*, Society of Petroleum Engineers, Dallas, 1985, pp. 415–428.
17. J. R. Appleyard, I. M. Cheshire and R. K. Pollard, 'Special techniques for fully-implicit simulators', presented at the *European Symposium on Enhanced Oil Recovery*, Bournemouth, U.K., September 1981.
18. U. R. B. Obeysekare, 'A two dimensional model for the immiscible displacement of oil using the Buckley–Leverett approach', *M.S. Thesis*, Department of Petroleum Engineering, University of Wyoming, Laramie, Wyoming, 1985.
19. J. W. Mercer and C. R. Faust, 'The application of finite element techniques to immiscible flow in porous media', in W. G. Gray *et al.* (eds), *Finite Elements in Water Resources*, Pentech Press, London, 1977, pp. 21–43.
20. R. B. Lantz, 'Quantitative evaluation of numerical diffusion (truncation error)', *Soc. Pet. Eng. J.*, September, 1971, pp. 315–320.
21. M. B. Allen, 'Numerical modelling of multiphase flow in porous media', *Adv. Water Resources*, **8**, 162–187 (1985).
22. R. E. Ewing, 'Problems arising in the modeling of processes for hydrocarbon recovery', in R. E. Ewing (ed), *The Mathematics of Reservoir Simulation*, Society for Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1983, pp. 3–34.