
ELSEVIER Applied Mathematics and Computation 125 (2002) 347–358

APPLIED MATHEMATICS AND COMPUTATION

www.elsevier.com/locate/amc

On the performance of certain direct and iterative methods on equations arising on a two-dimensional in situ combustion simulator

A. Refik Bahadir [a,*], F. Brian Ellerby [b]

[a] *Department of Mathematics, Faculty of Arts and Sciences, İnönü University, 44100 Malatya, Turkey*

[b] *School of Mathematical Sciences, University of Bath, Bath, UK BA2 7AY*

Abstract

A two-dimensional mathematical model of the in situ combustion process involves a set of nonlinear partial differential equations. These equations are discretized in implicit finite-difference form. The resulting set of nonlinear algebraic equations are solved for each time-step by use of a Newton–Raphson procedure. Each Newton iteration produces an equation of the form

$$A\mathbf{x} = \mathbf{b}, \qquad (*)$$

where $\mathbf{x}$ is the Newton update, $\mathbf{b}$ is the current residual of the nonlinear equations and A is the Jacobian matrix. A is large and has a non-symmetric, sparse structure. In this current work we wish to compare the performance of LU factorization, ORTHOMIN(m) and more recent iterative methods, GMRES(m) and BI-CGSTAB to solve $(*)$ on the model of the in situ combustion problem. To increase the convergence rate for the iterative methods a preconditioning and a scaling technique are used.

Keywords: In situ combustion; GMRES; ORTHOMIN; BI-CGSTAB

* Corresponding author.
E-mail address: arbahadir@inonu.edu.tr (A.R. Bahadir).

0096-3003/02/$ - see front matter
PII: S0096-3003(00)00136-3

1. Introduction

In situ combustion is one of the oil recovery methods in which thermal energy is used to increase oil recovery. The process requires the burning of some reservoir oil in place. Heat is generated to produce a rise in the reservoir temperature and a reduction in the viscosity of the oil: hence, the oil flows freely towards the production well.

The model consists of nonlinear partial differential continuity equations which govern the multiphase flow of gas, oil and water and the energy equation in the reservoir medium.

The discretization in time is fully-implicit and central-difference methods combined with one-point upstream weighting are employed for space discretization. The discretization yields coupled nonlinear algebraic equations per grid block.

The resulting system of nonlinear finite difference equations are solved by the Newton–Raphson method. The linear system of equations implicit in the case of Newton's method was solved by using *direct* or *iterative* methods at each step of the Newton's iteration. Direct methods (like Gaussian Elimination and LU factorization) may be inefficient for solving an equation of the form

$$A\mathbf{x} = \mathbf{b} \tag{1}$$

(where $\mathbf{x}$ is the Newton update, $\mathbf{b}$ is the current residual of the nonlinear equations and A is the Jacobian matrix. A is large and has a non-symmetric, sparse structure) because of the large amount of work and storage involved. Iterative methods can be used instead to obtain an approximate solution.

In Section 4, we illustrate the validity and efficiency of iterative methods for solving large linear systems arising from the finite difference discretizations of the equation governing the in situ combustion process. In particular, we consider the Generalized Minimal Residual Method (GMRES(m)) [12], the Orthogonal Minimization Method (ORTHOMIN(m)) [15] and the Biconjugate Gradient Stabilized Method (BI-CGSTAB) [14] Scaling and incomplete factorization preconditioning of the linear equations are used to improve the convergence properties of the iterative methods.

2. Model description and computational procedure

The set of the simulator equations is composed of five mass conservation equations and an energy conservation equation [2,3]. For simplicity we will consider the equations in the following non-dimensionalized form.

The mass conservation equation:

$$\frac{\partial}{\partial x}\left(P_r\frac{\partial a}{\partial x}\right)-\frac{\partial}{\partial x}(Q_r)+\frac{\partial}{\partial y}\left(P_r\frac{\partial a}{\partial y}\right)-\frac{\partial}{\partial y}(R_r)+S_r=\frac{\partial}{\partial t}(T_r), \tag{2}$$

where $r=1,2,3,4,5$ for water, heavy oil, light oil, inert gas and oxygen mass conservation.

The energy conservation equation:

$$\lambda\frac{\partial^2 b}{\partial x^2}+\frac{\partial}{\partial x}\left(U\frac{\partial a}{\partial x}\right)-\frac{\partial}{\partial x}(V)+\lambda\frac{\partial^2 b}{\partial y^2}+\frac{\partial}{\partial y}\left(U\frac{\partial a}{\partial y}\right)-\frac{\partial}{\partial y}(X)+Y=\frac{\partial}{\partial t}(Z), \tag{3}$$

where λ is the thermal conductivity, $P_r, Q_r, S_r, R_r, T_r, U, V, X, Y, Z$ are functions of a, b, c, d, e, g and a, b, c, d, e, g are functions of x, y, t.

Boundary conditions are considered to be no flow of mass or energy across the reservoir boundaries. This is done by setting the depending variables to zero in the boundary grid blocks.

The initial conditions for the depending variables are available as constant values according to the reservoir conditions [2].

The numerical technique of *finite differences*, is used to solve Eqs. (2) and (3). The equations are discretized using central differences with combined upstream weighting for the spatial variables and backward differences in time [1].

The discretization of the mass conservation equations

$$\begin{aligned}&\frac{1}{\Delta x^2}\left[(P_r)^{n+1}_{i,j}\left(a^{n+1}_{i+1,j}-a^{n+1}_{i,j}\right)-(P_r)^{n+1}_{i-1,j}\left(a^{n+1}_{i,j}-a^{n+1}_{i-1,j}\right)\right]-\frac{1}{\Delta x}\left[(Q_r)^{n+1}_{i,j}\right.\\&\left.-(Q_r)^{n+1}_{i-1,j}\right]+\frac{1}{\Delta y^2}\left[(P_r)^{n+1}_{i,j}\left(a^{n+1}_{i,j+1}-a^{n+1}_{i,j}\right)-(P_r)^{n+1}_{i,j-1}\left(a^{n+1}_{i,j}-a^{n+1}_{i,j-1}\right)\right]\\&-\frac{1}{\Delta y}\left[(R_r)^{n+1}_{i,j}-(R_r)^{n+1}_{i,j-1}\right]+(S_r)^{n+1}_{i,j}\\&=\frac{1}{\Delta t}\left[(T_r)^{n+1}_{i,j}-(T_r)^{n}_{i,j}\right].\end{aligned}$$

The discretization of the energy conservation equation

$$\begin{aligned}&\frac{\lambda}{\Delta x^2}\left(b^{n+1}_{i-1,j}-2b^{n+1}_{i,j}+b^{n+1}_{i+1,j}\right)+\frac{\lambda}{\Delta y^2}\left(b^{n+1}_{i,j-1}-2b^{n+1}_{i,j}+b^{n+1}_{i,j+1}\right)\\&+\frac{1}{\Delta x^2}\left[U^{n+1}_{i,j}\left(a^{n+1}_{i+1,j}-a^{n+1}_{i,j}\right)-U^{n+1}_{i-1,j}\left(a^{n+1}_{i,j}-a^{n+1}_{i-1,j}\right)\right]-\frac{1}{\Delta x}\left(V^{n+1}_{i,j}-V^{n+1}_{i-1,j}\right)\\&+\frac{1}{\Delta y^2}\left[U^{n+1}_{i,j}\left(a^{n+1}_{i,j+1}-a^{n+1}_{i,j}\right)-U^{n+1}_{i,j-1}\left(a^{n+1}_{i,j}-a^{n+1}_{i-1,j}\right)\right]-\frac{1}{\Delta y}\left(X^{n+1}_{i,j}-X^{n+1}_{i,j-1}\right)\\&+Y^{n+1}_{i,j}=\frac{1}{\Delta t}\left(Z^{n+1}_{i,j}-Z^{n}_{i,j}\right).\end{aligned}$$

The discretization leads to six nonlinear algebraic equations for each grid block. The set of nonlinear algebraic equations is solved using Newton's method. For each grid block the equations are written as:

$$f_1(W) = 0,$$
$$f_2(W) = 0,$$
$$f_3(W) = 0,$$
$$f_4(W) = 0,$$
$$f_5(W) = 0,$$
$$f_6(W) = 0$$

or

$$F(W) = \mathbf{0}, \tag{4}$$

where $F = ((f_1)_1, \ldots, (f_6)_1, \ldots, (f_1)_N, \ldots, (f_6)_N)$, N is the number of grid blocks and $W = (a, b, c, d, e, g)$.

Then to solve the nonlinear equation system (4) Newton's method can be written as:

$$J(W^k)\Delta^{k+1} = -F(W^k),$$
$$W^{k+1} = W^k + \Delta^{k+1}.$$

This is applied iteratively at each time step. At each new time, the initial guess for the Newton's iteration is the solution at the previous time.

The matrix $J(W^k)$ is the Jacobian matrix and takes the five block diagonal form. It also has a non-symmetric, sparse structure.

At the each step of Newton's iteration, the linear system of equations

$$A\mathbf{x} = \mathbf{b},$$

where $A = J(W^k)$, $\mathbf{x} = \Delta^{k+1}$ and $\mathbf{b} = -F(W^k)$.

This can be done by using direct or iterative methods.

3. Methods for solving linear system of equations

In a direct method, as the name implies, the algorithm guaranteed to lead to the exact solution after a finite number of mathematical operations in the absence of round-off errors. Because of the large order of most sparse systems of linear equations the linear system cannot usually be solved efficiently by a direct method. Iterative methods are the preferred method of solution.

Iterative methods are commonly used to solve problems resulting from the discretizations of couple partial differential equations, and time-dependent problems involving more than one spatial variable. The important advantages

of iterative methods are the simplicity and uniformity of the operations to be performed, which make them well suited for use on computers, also they are relatively insensitivity to round-off errors.

Recently, many iterative methods have been derived for solving non-symmetric linear systems of equations. Some outstanding examples are the ORTHOMIN(m) [15], the GMRES(m) [12], the Generalized Conjugate Gradient Method [5,16], the Generalized Conjugate Residual Method [6], the Orthogonal Residual Method (ORTHORES) [17], the Biconjugate Gradient Method (BCG) [7,10], the Conjugate Gradient Squared Method (CGS) [13] and the BI-CGSTAB [14].

3.1. Comparison direct and iterative methods on the in situ combustion simulator

In general, the coefficient matrices in reservoir simulation problems are block matrices and they are also sparse. For small size problems direct solution may well be an efficient method. For example, for 1D models less than 15 grid blocks in length, direct solution will be the preferred method. But the simulation of in situ combustion process in 2D or 3D requires the use of large numbers of gridblocks for accurate models as a result of this sets of large, sparse linear system must be solved. It may will be more economical to solve such a system with iterative rather than direct methods.

However, these matrices are neither symmetric nor necessarily diagonally dominant. Particularly, the in situ combustion problem represents one of the most difficult types of problem that is encountered in simulation. It produces a highly non-symmetric matrix because of multiple moving fronts that can change their relative positions during the process [4].

In recent years, experience has shown that iterative methods such as ORTHOMIN [15] and ORTHORES [17] seem to be very robust for the types of problems met in reservoir simulation.

For comparison we consider LU factorization as a direct method and iterative methods. The iterative methods of interest here are the ORTHOMIN(m) which was developed by Vinsome [15]: it is a generalized conjugate residual method. GMRES(m) is the restarted version of the generalized minimal residual method of Saad and Schultz [12]. BI-CGSTAB is a one of the latest iterative method developed by Van der Vorst [14].

To increase the convergence rate for the iterative methods a preconditioning technique, Incomplete LU factorization (ILU), is used.

3.1.1. The Generalized Minimal Residual Method (GMRES(m))

The GMRES method begins with an initial approximation $\mathbf{x}_0$ and initial residual $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$, at the mth iteration, a correction $\mathbf{z}_m$ is determined in the Krylov subspace $\mathscr{K}_m(\mathbf{r}_0, A)$ which solves the least squares problem

$$\min_{\mathbf{z}\in\mathscr{K}_m(\mathbf{r}_0,A)} \|\mathbf{b}-A(\mathbf{x}_0+\mathbf{z})\|_2.$$

The mth iterate is then $\mathbf{x}_m=\mathbf{x}_0+\mathbf{z}_m$.

Algorithm 1 (*Preconditioned GMRES*(m)).

1. Choose $\mathbf{x}_0$.
2. Compute $\mathbf{r}_0=\mathbf{b}-A\mathbf{x}_0$.
3. Compute $\beta=\|\mathbf{r}_0\|_2$.
4. Compute $\mathbf{v}_1=\mathbf{r}_0/\beta$.
5. For $k=0,1,2,\ldots,$ MAXIT do:
   For $j=1,2,3,\ldots,m$ do:
   $\mathbf{z}=M^{-1}\mathbf{v}_j$
   $\mathbf{w}=A\mathbf{z}$
   For $i=1,2,3,\ldots,j$ do:
   $h_{i,j}=(\mathbf{w},\mathbf{v}_i)$
   $\mathbf{w}=\mathbf{w}-h_{i,j}\mathbf{v}_i$
   Enddo.
   $h_{j+1,j}=\|\mathbf{w}\|_2$
   $\mathbf{v}_{j+1}=\mathbf{w}/h_{j+1,j}$
   Enddo.
6. Form the approximate solution:
   $\mathbf{x}_{k+1}=\mathbf{x}_k+M^{-1}V_m\mathbf{y}_m$,
   where $\mathbf{y}_m$ minimizes $\|\beta\mathbf{e}_1-\hat{H}_m\mathbf{y}\|_2$, $\mathbf{y}\epsilon R^m$.
7. Compute $\mathbf{r}_{k+1}=\mathbf{b}-A\mathbf{x}_{k+1}$
8. Compute $\|\mathbf{r}_{k+1}\|_2$: if satisfied then stop.
9. Else: set $\beta=\|\mathbf{r}_{k+1}\|_2$
10. Compute $\mathbf{v}_1=\mathbf{r}_{k+1}/\beta$
    Enddo.

3.1.2. The orthogonal minimization method (ORTHOMIN(m))

For symmetric positive definite problems, the conjugate residual method computes a sequence of approximate solutions by an iteration of the form

$$\mathbf{x}_{k+1}=\mathbf{x}_k+\alpha_k\mathbf{p}_k,$$

and the step-length α_k minimizes

$$\|\mathbf{b}-A(\mathbf{x}_k+\alpha_k\mathbf{p}_k)\|_2=\|\mathbf{r}_{k+1}\|_2,$$

as a function of α_k. The direction vectors $\mathbf{p}_k$ are computed by a two term recurrence of the form

$$\mathbf{p}_{k+1}=\mathbf{r}_{k+1}+\beta_k\mathbf{p}_k,$$

and they satisfy

$$(A\mathbf{p}_k, A\mathbf{p}_j) = 0, \quad k \neq j. \tag{5}$$

In the non-symmetric case, a set of directions that satisfy (5) can be computed as follows,

$$\mathbf{p}_{k+1} = \mathbf{r}_{k+1} + \sum_{j=\max(0,k-m+1)}^{k} \beta_j^k \mathbf{p}_j,$$

where

$$\beta_j^k = -(A\mathbf{r}_{k+1}, A\mathbf{p}_j)/(A\mathbf{p}_j, A\mathbf{p}_j), \quad j \leqslant k. \tag{6}$$

This method is called as ORTHOMIN(m).

Algorithm 2 (*Preconditioned ORTHOMIN*(m)).

1. Choose $\mathbf{x}_0 = \mathbf{0}$.
2. Compute $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
3. Set $\mathbf{p}_0 = \mathbf{r}_0$.
4. For $k = 0, 1, 2, \ldots,$ MAXIT do:
   - $\alpha_k = (\mathbf{r}_k, A\mathbf{p}_k)/(A\mathbf{p}_k, A\mathbf{p}_k)$
   - $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k$
   - $\mathbf{r}_{k+1} = \mathbf{r}_k - \alpha_k A\mathbf{p}_k$
   - Compute $\|\mathbf{r}_{k+1}\|_2$; if satisfied then stop.
   - $\mathbf{z} = M^{-1}\mathbf{r}_{k+1}$
   - $j_k = \max(0, k - m + 1)$
   - For $j = j_k, \ldots, k$ do:
     - $\beta_j^k = -(A\mathbf{z}, A\mathbf{p}_j)/(A\mathbf{p}_j, A\mathbf{p}_j)$
   - Enddo.
   - $\mathbf{p}_{k+1} = \mathbf{z} + \sum_{j=j_k}^{k} \beta_j^k \mathbf{p}_j$
   - $A\mathbf{p}_{k+1} = A\mathbf{z} + \sum_{j=j_k} \beta_j^k A\mathbf{p}_j$

   Enddo.

3.1.3. The Biconjugate Gradient Stabilized Method (BI-CGSTAB)

The BI-CGSTAB is another Krylov subspace method, which has recently been developed by Van der Vorst [14] for solving non-symmetric linear systems. It is closely related to the BCG and the CGS. The main idea is to form a product of the BCG polynomial with another, locally defined polynomial. It is a more smoothly converging variant of CGS.

Algorithm 3 (*Preconditioned BI-CGSTAB*).

1. Choose $\mathbf{x}_0$.
2. Compute $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0$.
3. Set $\tilde{\mathbf{r}}_0 = \mathbf{r}_0$
   - $\mathbf{v}_0 = \mathbf{p}_0 = \mathbf{0}$

$\rho_0 = \omega_0 = \alpha = 1$

4. For $k = 1, 2, 3, \ldots,$ MAXIT do:

$\rho_k = (\tilde{\mathbf{r}}_0, \mathbf{r}_{k-1})$

$\beta = (\rho_k/\rho_{k-1})(\alpha/\omega_{k-1})$

$\mathbf{p}_k = \mathbf{r}_{k-1} + \beta(\mathbf{p}_{k-1} - \omega_{k-1}\mathbf{v}_{k-1})$

$\mathbf{y} = M^{-1}\mathbf{p}_k$

$\mathbf{v}_k = A\mathbf{y}$

$\sigma_k = (\tilde{\mathbf{r}}_0, \mathbf{v}_k)$

$\alpha = \rho_k/\sigma_k$

$\mathbf{s} = \mathbf{r}_{k-1} - \alpha\mathbf{v}_k$

$\mathbf{z} = M^{-1}\mathbf{s}$

$\mathbf{t} = A\mathbf{z}$

$\omega_k = (\mathbf{t}, \mathbf{s})/(\mathbf{t}, \mathbf{t})$

$\mathbf{x}_k = \mathbf{x}_{k-1} + \alpha\mathbf{y} + \omega_k\mathbf{z}$

$\mathbf{r}_k = \mathbf{s} - \omega_k\mathbf{t}$

Compute $\|\mathbf{r}_k\|_2$: if satisfied then stop.

Enddo.

3.2. Preconditioning and scaling

The rate of convergence of the iterative methods can usually be speeded up by the technique of preconditioning.

Let M be a non-singular matrix, then the basic principle of preconditioning is that we replace the original linear system (1) by the equivalent system

$$M^{-1}A\mathbf{x} = M^{-1}\mathbf{b},$$

where M^{-1} is a matrix such that $M^{-1}\mathbf{w}$ is inexpensive to compute for any vector $\mathbf{w}$.

ILU factorization is one of the most popular preconditioning technique for the iterative methods. It has been successfully used by many authors in the symmetric positive definite case [8]. It appears also very attractive in the non-symmetric case, as reported in [6,9,11].

3.2.1. Incomplete LU factorization (ILU)

During the steps of a direct solution procedure additional non-zero elements are generated. These are called fill-in terms. Fill-in increases storage requirements and computational work. Because of fill-in, Matrices L and U in the factorization $A = LU$ are much less sparse than the original matrix A.

The basic idea of *ILU* factorization is to preserve the sparsity in the L and U factors by neglecting most of, or even all, the fill-in terms during the elimination procedure. Of course, the product LU is then only an approximation of A, i.e.

$$A \approx M = LU,$$

where L is a lower triangular matrix and U is an upper triangular matrix.

ILU preconditioning has been used here for all the iterative schemes and is defined as follows: Let $\mathcal{N} \subseteq \{(i,j) \mid 1 \leqslant i,j \leqslant n\}$. In our numerical experiments $\mathcal{N}$ contains the indices of the elements on the five blocks diagonal.

Algorithm 4 (*Incomplete LU factorization*).

For $i = 1,2,3,\ldots,n$ do:
  For $j = 1,2,3,\ldots,n$ do:
    If $(i,j) \in \mathcal{N}$ then
      $s_{ij} = a_{ij} - \sum_{k=1}^{min(i,j)-1} l_{ik}u_{kj}$
      If $i \geqslant j$ then $l_{ij} = s_{ij}$
      If $i < j$ then $u_{ij} = s_{ij}/l_{i,i}$
    Endif.
  Enddo.
  $u_{ii} = 1$
  For $j = i+1, i+2, i+3, \ldots, n$ do:
    $u_{ij} = u_{ij}/l_{ii}$
  Enddo.
Enddo.

A scaling of the set of equations is also tested in the system. The approach is to define

$$s_i = \max_{1 \leqslant j \leqslant n} |a_{ij}|, \quad i = 1,2,3,\ldots,n,$$

$$\hat{a}_{i,j} = a_{ij}/s_i, \quad i,j = 1,2,3,\ldots,n,$$

$$\hat{b}_i = b_i/s_i, \quad i = 1,2,3,\ldots,n,$$

where $\hat{A} = [\hat{a}_{ij}]$ and $\hat{b} = [\hat{b}_i]$ are the result of scaling.

The convergence criterion imposed on the residual was varied from the one-dimensional case to two-dimensional depending on the value of the residual calculated when using the direct method. The object was to make comparable results which are obtained by direct and iterative methods. All the iterative methods use a vector of zeros as an initial guess.

4. Numerical results

In this section, we describe the results of numerical experiments in which the methods given above are used. A number of test were performed to compare *LU* factorization, GMRES, ORTHOMIN and BI-CGSTAB. We performed the computations on various grid blocks: 4×4, 8×8 and 12×12. Tables 1–3

show the results for LU factorization, BI-CGSTAB, GMRES(m) and ORTHOMIN(m) with various values for m.

The flags used in the tables are defined as follows:

RT: Reservoir Time (s),

LIPTS: Linear Iterations Per Time Step,

NIPTS: Newton Iterations Per Time Step,

LIPNI: Linear Iterations Per Newton Iteration,

TOTLI: Total Linear Iterations,

TOTNI: Total Newton Iterations,

WORK: Work Required for All Time Step,

We measure the work for each iteration in terms of the number of arithmetic operations, multiplication and division, required to perform it. We consider all iterative methods with preconditioning and scaling.

In the first problem a 4×4 grid block was used, so order of the matrix was 96. All methods reached the same reservoir time in five time steps and taken an average of 16.8 Newton iterations per time step except GMRES(1). It required average of 17.2 Newton iteration to reach the convergence.

As a result of the size of the problem, LU factorization required less work than all iterative methods. In generally, among the iterative methods GMRES, particularly GMRES(5), seems better than other two methods. The performance of GMRES(m) and ORTHOMIN(m) depends upon the chosen value of m. Both methods required same amount of linear iteration for $m = 10$ and $m = 20$ to reach the convergence. BI-CGSTAB is better than ORTHOMIN as shown in Table 1.

The second example is a 8×8 grid block case. Table 2 illustrates the typical convergence behaviour of each of the iterative methods. In this case iterative methods required less work than the direct method LU factorization however, BI-CGSTAB is better than ORTHOMIN and looks competitive with GMRES but GMRES(5) is the best of all.

Table 1
Results for the two-dimensional (4×4) simulation

Methods	RT	LIPTS	NIPTS	LIPNI	TOTLI	TOTNI	WORK
LU factorization	361	–	16.8	–	–	84	225×10^4
GMRES(1)	361	600.2	17.2	34.90	3001	86	211×10^4
GMRES(5)	361	72.4	16.8	4.31	362	84	314×10^4
GMRES(10)	361	72.6	16.8	4.32	363	84	331×10^4
GMRES(20)	361	72.6	16.8	4.32	363	84	366×10^4
ORTHOMIN(1)	361	362.6	16.8	21.58	1813	84	894×10^4
ORTHOMIN(5)	361	203.4	16.8	12.11	1017	84	652×10^4
ORTHOMIN(10)	361	166.6	16.8	9.92	833	84	668×10^4
ORTHOMIN(20)	361	166.6	16.8	9.92	833	84	908×10^4
BI-CGSTAB	361	96.2	16.8	5.73	481	84	487×10^4

Table 2
Results for the two-dimensional (8×8) simulation

Methods	RT	LIPTS	NIPTS	LIPNI	TOTLI	TOTNI	WORK
LU factorization	260	–	18.6	–	–	93	108×10^6
GMRES(1)	260	719.8	18.4	39.12	3599	92	114×10^6
GMRES(5)	260	154.0	18.6	8.28	770	93	331×10^5
GMRES(10)	260	169.4	18.6	9.11	847	93	370×10^5
GMRES(20)	260	169.8	18.6	9.13	849	93	403×10^5
ORTHOMIN(1)	260	625.0	18.6	35.05	3260	93	721×10^5
ORTHOMIN(5)	260	405.0	18.6	21.77	2025	93	583×10^5
ORTHOMIN(10)	260	3659.4	19.2	190.59	18 297	96	516×10^6
ORTHOMIN(20)	260	291.0	18.6	15.65	1455	93	701×10^5
BI-CGSTAB	260	177.6	18.6	9.55	888	93	426×10^5

Table 3
Results for the two-dimensional (12×12) simulation

Methods	RT	LIPTS	NIPTS	LIPNI	TOTLI	TOTNI	WORK
LU factorization	72	–	17.6	–	–	88	114×10^7
GMRES(1)	72	515.4	17.6	29.23	2572	88	222×10^6
GMRES(5)	72	118.6	17.6	6.74	593	88	930×10^5
GMRES(10)	72	129.2	17.6	7.34	646	88	993×10^5
GMRES(20)	72	129.8	17.6	7.38	649	88	105×10^6
ORTHOMIN(1)	72	588.0	17.6	33.41	2940	88	180×10^6
ORTHOMIN(5)	72	422.6	17.6	24.01	2113	88	166×10^6
ORTHOMIN(10)	72	2032.0	16.1	126.12	18 288	145	130×10^7
ORTHOMIN(20)	72	1946.8	18.4	105.80	9734	92	949×10^6
BI-CGSTAB	72	197.8	17.6	11.24	989	88	134×10^6

Finally, we can see from Table 3 that for a 12×12 grid block (i.e. order of the matrix was 864) the advantage of iterative methods is very clear when we consider the amount of work which they required. In this example ORTHOMIN performed far less well than GMRES and BI-CGSTAB, especially when 10 and 20 orthogonal vectors were used. The performances of GMRES for $m = 5$ and $m = 10$ are very close but, GMRES(5) is cheaper than the others.

5. Conclusion

From the above results, if the system is large then iterative methods GMRES, ORTHOMIN and BI-CGSTAB provide a valuable tool for solving linear system arising from simulation of the in situ combustion. Among the iterative methods ORTHOMIN seems to be less robust and required more work than others. BI-CGSTAB performed better than ORTHOMIN but is not

competitive to GMRES and particularly GMRES(5) is certainly the cheapest of the methods tried here.

References

[1] K. Aziz, A. Settari, Petroleum Reservoir Simulation, Applied Science Publishers, London, 1979.
[2] A.R. Bahadir, Two-dimensional computer simulation of in situ combustion oil recovery process and iterative methods, Ph.D. Thesis, The University of Bath, 1994.
[3] A.R. Bahadir, F.B. Ellerby, M. Greaves, A two-dimensional in situ combustion simulator, In-Situ (under review).
[4] A. Behie, P.K.W. Vinsome, Block iterative methods for fully implicit reservoir simulation, in: Paper SPE 9303, Presented at the 55th Annual Fall Technical Conference, Dallas, September, 1982.
[5] P. Concus, G.H. Golub, A generalized conjugate gradient method for nonsymmetric systems of linear equations, in: R. Glowinski, J.L. Lions (Eds.), Lecture Notes in Economics and Mathematical Systems, vol. 134, Springer, Berlin, 1976.
[6] H.C. Elman, Iterative methods for large, sparse, nonsymmetric systems of linear equations, Ph.D. Thesis, Yale University, New Haven, CT, 1982.
[7] R. Fletcher, Conjugate gradient methods for indefinite systems, in: G.A. Watson (Ed.), in: Proceedings of the Dundee Biennial Conference on Numerical Analysis, Springer, New York, 1975.
[8] D.S. Kershaw, The incomplete Cholesky-conjugate gradient method for the iterative solution of systems of linear equations, J. Comput. Phys. 26 (1978) 43–65.
[9] A.E. Koniges, D.V. Anderson, ILUBCG2: a preconditioned biconjugate gradient routine for the solution of linear asymmetric matrix equations arising from 9-point discretizations, Comput. Phys. Commun. 43 (1987) 297–302.
[10] C. Lanczos, Solution of systems of linear equations by minimized iterations, J. Res. Nat. Bur. Standards 49 (1952) 33–53.
[11] Z. Mikic, E.C. Morse, The use of a preconditioned bi-conjugate gradient method for hybrid plasma stability analysis, J. Comput. Phys. 61 (1985) 154–185.
[12] Y. Saad, M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 7 (1986) 856–869.
[13] P. Sonneveld, CGS, a fast Lanczos-type solver for nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 10 (1989) 36–52.
[14] H.A. van der Vorst, BI-CGSTAB: a fast and smoothly converging variant of BI-CG for the solution of nonsymmetric linear systems, SIAM J. Sci. Statist. Comput. 13 (1992) 631–644.
[15] P.K.W. Vinsome, Orthomin, an iterative method for solving sparse sets of simultaneous linear equations, Paper SPE 5729, in: Proceedings Fourth Symposium on Reservoir Simulation, SPE of AIME, Los Angeles, February, 1976.
[16] O. Widlund, A Lanczos method for a class of nonsymmetric systems of linear equations, SIAM J. Numer. Anal. 15 (1978) 801–812.
[17] D.M. Young, K.C. Jea, Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods, Linear Algebra Appl. 34 (1980) 159–194.