SPE 115017

Solving Linear Equations In Reservoir Simulation Using Multigrid Methods

R. P. Hammersley, SPE, and D. K. Ponting, SPE, Roxar

This paper was prepared for presentation at the 2008 SPE Russian Oil & Gas Technical Conference and Exhibition held in Moscow, Russia, 28–30 October 2008.

Abstract

The sets of linear equations which arise in large scale reservoir simulation can present problems for traditional linear solution algorithms. These scale poorly as the number of cells increases, and convergence is limited by the use of permeability distributions from geological modelling, with correlated high permeability channels, and the modelling of long horizontal wells. In addition, there is a requirement that the solver runs well on computers with many parallel processors.

The use of multi-level methods offers the possibility of good scaling with problem size. A common approach has been to use an algebraic multigrid (AMG) method. The integration of such a technique into a parallel reservoir simulation code is described, and the performance of different coarsening and interpolation algorithms compared.

AMG methods generally perform well when solving for a scalar solution to a system of elliptic equations and are not as straightforward to generalize to vector solution variables as traditional grid-based solvers. However, in fully implicit reservoir simulation a solution change in terms of both pressures and saturations or molar densities is generally required. This has lead to the use of solvers which combine an AMG step with a secondary solver step. There can be advantages in selecting non-linear variables and equation sets to suit this technique, rather than treating the linear solver as a self-contained black box. Such methods are described and compared with traditional methods for black oil and compositional simulation.

1. Introduction

The solution of large sets of sparse, non-symmetric linear equations is at the core of reservoir simulation. Over the last 25 years, the most commonly used methods have been incomplete lower-upper decompositions with various levels of retained fill-in, ILU(n) (ref. 1); nested factorisation (ref. 2) and constrained residual (CPR) methods, (refs. 3 and 4). These solvers are generally used as preconditioners within an ORTHOMIN (ref. 5) or a GMRES (ref. 6) acceleration scheme. More recently, there has been interest in multilevel solvers, and in particular algebraic multigrid (AMG) techniques. These offer good asymptotic scaling of solution effort with problem size, and a solver which automatically adapts to features in the problem such as channels, faults, horizontal wells and flexible gridding.

AMG solvers are most efficient in solving elliptic problems in which level by level smoothing is highly effective. Reservoir simulation is frequently performed in a fully implicit mode, in which both pressures and saturations or molar densities at the end of the time step are solved for simultaneously. Saturations may exhibit sharp fronts which are less suited to solution by coarse grid corrections. A common method, and one used here, is to use an AMG solver as part of a two-stage preconditioning (refs. 7, 8, 9, 10 and 11), the AMG solver being applied to generate a pressure correction only. The extraction of a suitable set of pressure equations is discussed in section 2.

The AMG solver used to solve the resulting pressure equations is described in section 3. This involves three main elements: a coarsening strategy, an interpolation algorithm and a relaxation operator. Two coarsening options are detailed, together with two interpolation algorithms. A set of data structures which assist in implementing the AMG solver in parallel is then described.

For IMPES (implicit pressure, explicit saturation) simulations, only pressure unknowns are solved for at each non-linear iteration, and the AMG pressure solver is sufficient. For the fully implicit case, however, we need to find changes in pressures and in saturations or molar densities. To do this a second stage is employed (refs. 4, 7, 8, 9, 10, 11 and 12), an ILU(0) method which solves for both pressures and the saturation or molar variable changes simultaneously. The need for a second pressure solve may seem surprising, as a set of pressure changes was provided by the AMG step – however, this does not prove effective in terms of an overall solution, and the combination of two stages has proved better overall. The search directions from the two solver stages may be provided in sequence to the surrounding GMRES algorithm, or a simple combination can be used by adding the two search directions. Tests indicate that the simple combination is best – this may suggest that the approximations involved in setting up a pressure equation yield a pressure change which is good in terms of the inclusion of long range effects, but which is not a precise solution to the overall problem. The second stage is required to correct for short range effects within the context of a fully implicit method. Setting up a pressure equation on the basis of diagonal pivoting includes some upstreamed flow derivatives in the Jacobian but not others: an alternative is described in which an IMPES type correction is applied before entering the linear solver on the basis of the mass accumulation part of the non-linear residual only. Finally some test results are included and development possibilities discussed.

2. Solving the linear equations

The non-linear equations which arise in reservoir simulation are normally solved using Newton-Raphson iteration. At each step of the solution a non-linear residual function R is obtained, the step is solved when a solution vector X is calculated such that $R(X)=0$. This is done by repeatedly calculating the Jacobian $J=\partial R/\partial X$, where X is the current solution vector which defines the current state of the reservoir and solving for $\Delta X=-J^{-1}R(X)$, $X:=X+\Delta X$ until $R(X)$ meets the required convergence criteria. If the number of active cells in the study is N_a, then the Jacobian J is a sparse $N_a \times N_a$ matrix. For a regular 3-dimensional grid the basic pattern of J would be a septa-diagonal matrix reflecting the flow coupling of each cell to its 6 neighbours in the grid; in practice faults and wells contribute other coupling terms, but the overall matrix remains sparse. For simulations run in IMPES mode each element of the Jacobian is a single number and the matrix can be inverted using an AMG solver only. However, in the fully implicit case the situation is more complex and each element of J is an $N \times N$ sub-matrix, for a system with N mass conservation equations for each cell. The N primary variables are typically a reference pressure, p, and $N-1$ saturations, s, in the black oil case.

The linear equations to be solved may be cast into the conventional form $A.x=b$ with $A=J$, $x=\Delta X$ and $b=-R$. Each element of the matrix in the fully implicit case may be written as:

$$A_{ij}=\left(\begin{array}{c|ccc} \dfrac{\partial R_{1i}}{\partial P_j} & \dfrac{\partial R_{1i}}{\partial S_{1j}} & \cdots & \dfrac{\partial R_{1i}}{\partial S_{N-1j}} \\ \hline \vdots & \vdots & \ddots & \\ \dfrac{\partial R_{Ni}}{\partial P_j} & \dfrac{\partial R_{Ni}}{\partial S_{1j}} & \cdots & \dfrac{\partial R_{Ni}}{\partial S_{N-1j}} \end{array}\right)=\left(\begin{array}{c|c} \left(A_{1p}\right)_{ij} & \left(A_{1s}\right)_{ij} \\ \hline \left(A_{cp}\right)_{ij} & \left(A_{cs}\right)_{ij} \end{array}\right) \qquad \begin{array}{l} s=1\ldots N-1 \\ c=2\ldots N \\ i,j=1\ldots N_a \end{array}$$

where N_a is the number of grid cells. $(A_{1p})_{ij}$ is a 1×1 matrix, $(A_{1s})_{ij}$ is a $1\times(N-1)$ matrix, $(A_{1s})_{ij}$ is a $(N-1)\times 1$ matrix and $(A_{cs})_{ij}$ is an $(N-1)\times(N-1)$ matrix.

This section will discuss the construction of an efficient iterative scheme to solve these linear equations and follows ref. 11. The variables of pressure and saturation have different behavior in the mass conservation equations: pressure acts in a long-range manner influencing large portions of the reservoir, whilst the saturations act in a short-range manner. This suggests that the two variable sets, pressure and saturations, may be solved using two different types of solver, one which is effective for long-range variables and the other for short-range variables. However, it is not generally simple to decouple the system into two equation sets which can be treated in this way.

Consider the first equation, which can be written as:

$$A_{1p}x_p + A_{1s}x_s = b_1$$

where $x_p=(x_{p1},\ldots,x_{pN_a})$ is the set of pressure changes and $x_s=(x_{s\,1\,1},\ldots,x_{s\,N-1\,1},\ldots,x_{s\,1\,N_a},\ldots,x_{s\,N-1\,N_a})$ is the set of saturation changes for all the cells in the problem.

It can be seen that the saturation variables influence the pressure variable through A_{1s}. If A_{1s} was identically zero then it would be possible to solve for x_p directly using $A_{1p}x_p=b_1$ and then for x_s using a back substitution into:

$$A_{cs}x_s = b_s - A_{cp}x_p$$

This method could be realized if a transformation of the equations was made which sets A_{1s} to zero. Algebraically, this can be achieved using the Schur complement which may be written as:

$$\begin{pmatrix} I & -Q \\ & I \end{pmatrix}\begin{pmatrix} A_{1p} & A_{1s} \\ A_{cp} & A_{ss} \end{pmatrix} = \begin{pmatrix} A_{1p} - QA_{cp} & A_{1s} - QA_{cs} \\ A_{cp} & A_{cs} \end{pmatrix}$$

where $Q = A_{1s}A_{cs}^{-1}$, which gives $A_{1s} - QA_{cs} = A_{1s} - A_{1s}A_{cs}^{-1}A_{cs} = 0.$

However, these are large matrices which means it is difficult to form A_{cs}^{-1} and even if it could be formed would result in a dense matrix which would destroy the sparsity pattern of the equations needed for realistic solve times. What is required is a matrix Q which reduces the effect of the coupling matrix A_{1s} and maintains the sparsity pattern. This can be partially achieved by using the diagonal entries of the various matrices. Let A denote a block matrix, with block dimensions $l \times m$, then let D denote the block diagonal of the matrix in the sense of cell indices:

$$D(A)_{ij} = \begin{cases} (A)_{ii} & \text{for } i = j \\ 0 & \text{for } i \neq j \end{cases}$$

If the effect of flow terms were small, the coupling matrix A_{1s} might be reasonably well approximated by its diagonal term and this would suggest taking the block diagonal matrix:

$$Q_{1c} = \sum_s D(A_{1s})D(A_{cs})^{-1}$$

where $D(A_{1s})$ is a $1 \times (N-1)$ block matrix and $D(A_{cs})^{-1}$ is a $(N-1) \times (N-1)$ block matrix. This can be thought of as Gaussian elimination on the diagonal (ref. 12), ignoring the cell to cell flow coupling and corresponds to the Quasi-IMPES decoupling operator (ref. 9). The transformed linear equations take the form:

$$\begin{pmatrix} I & -D(A_{1s})D(A_{cs})^{-1} \\ & I \end{pmatrix}\begin{pmatrix} A_{1p} & A_{1s} \\ A_{cp} & A_{cs} \end{pmatrix} = \begin{pmatrix} A_{1p} - D(A_{1s})D(A_{cs})^{-1}A_{sp} & A_{1s} - D(A_{1s})D(A_{cs})^{-1}A_{cs} \\ A_{cp} & A_{cs} \end{pmatrix}$$
$$= \begin{pmatrix} \tilde{A}_{1p} & \tilde{A}_{1s} \\ A_{cp} & A_{cs} \end{pmatrix} = \tilde{A}$$

Note, the block diagonal of $\tilde{A}_{1s}$ is zero for $D(\tilde{A}_{1s}) = D(A_{1s}) - \sum_{s,c} D(A_{1s})D(A_{cs})^{-1}D(A_{cs}) = 0$, which was assumed to be the dominant term in the coupling matrix. The Quasi-IMPES decoupling operator also helps resolve the situation where the component corresponding to the first conservation equation is not present in a cell, as it mixes in all the different residual equations, one of which must have a component in the cell. Quasi-IMPES decoupling can be applied as a global left preconditioner, which adjusts the convergence factor, or it can be integrated as part of the pressure preconditioner in the manner of CPR. In this work, the former approach is taken.

Now the coupling matrix $\tilde{A}_{1s}$ has been weakened a combinative 2 stage pre-conditioner (ref. 12) can be used:

1. Make an approximate solution to $\tilde{A}_{1p}x_p = \tilde{b}_1$
2. Calculate the residual $r = b - \tilde{A}\begin{pmatrix} x_p \\ 0 \end{pmatrix}$
3. Make an approximate solution $\tilde{A}y = r$
4. Return $\begin{pmatrix} x_p \\ 0 \end{pmatrix} + y$ as an approximation to $\tilde{A}^{-1}\begin{pmatrix} \tilde{b}_1 \\ b_s \end{pmatrix}$

Finally, the two stage preconditioner is used iteratively in GMRES until the desired linear convergence factor is reached.

The above formulation has two distinctive features: it might be expected, that having formed a pressure equation and solved for x_p the second stage of the pre-conditioner would simply back substitute the pressure change and solve for the remaining variables. This technique does not work well in general. Instead, the whole system is re-solved for both x_p and x_s. Although the preconditioner is enclosed in a GMRES acceleration which can optimize the step length taken in terms of the reduction of the linear solver error, a simple linear combination of the search directions from each stage is used. This strongly suggests that the approximations made in setting up the pressure equation are quite severe, and that although the result of the AMG pressure solve can play a vital role in aiding convergence, it is not always a good pressure step in its own right.

3. The AMG Pressure solver

The first stage of the 2 stage preconditioner requires an approximate solution for the pressure equation, $\tilde{A}_{1p}x_p = \tilde{b}_1$. This equation is broadly elliptic and is amenable to AMG. This section will give brief descriptions of the main algorithms used, for more details on AMG the reader is referred to refs. 13 and 14. AMG is attractive as a solver, because it offers good scaling in model size and the number of processors. An AMG solver comprises two distinct parts, an analysis stage where the hierarchy of coarse matrices is constructed with their interpolation operators and a solve stage where the equations are actually solved. The analysis stage comprises two core algorithms, a coarsening strategy and an interpolation algorithm. The solve stage also comprises two core algorithms, matrix-vector multiplication and a relaxation algorithm.

Coarsening Strategy

An AMG solver builds a hierarchy of matrices and interpolation operators. The coarsening strategy at each level splits the variables into two types, a coarse set, *C*, and a fine set, *F*. The coarsening strategy is applied recursively when building the hierarchy, by using the coarse set as the variable set at the next coarsest level. The coarsening is then repeated until the number of variables has been reduced sufficiently that a direct solver can be used. In the process of making the coarse/fine split it is assumed the fine variables can be interpolated from the coarse variables. The interpolation algorithm used is highly dependent on the coarsening strategy chosen.

One of two coarsening strategies can be used. The first is CLJP (refs. 15 and 16), the second is PMIS (ref. 17). Given a matrix *A*, a variable *i* depends on *j* if a_{ij} is "large", for matrices with a positive diagonal, the set of dependencies of *i* is defined by:

$$S_i = \left\{ j \neq i : -a_{ij} \geq \alpha \max_{k \neq i} \left(-a_{ik} \right) \right\}$$

where α is a strength variable, typically set to 0.5 for a 3 dimensional problem. The set of variables which are influenced by *i* is given by $S_i^T = \left\{ j : i \in S_j \right\}$. The set of dependencies can be turned into an influence matrix, by:

$$S_{ij} = \begin{cases} 1 & \text{if } j \in S_i, \\ 0 & \text{otherwise.} \end{cases}$$

The CLJP algorithm proceeds as follows. Each variable *i* is assigned a weight $w(i) = \left| S_i^T \right| + \sigma(i)$, the number of variables influenced by *i* plus a random number from $(0,1)$. Now proceed recursively on all variables which have not been selected as either coarse or fine and define:

$$D = \left\{ i : w(i) > w(j) \text{ for all } j \text{ such that } S_{ij} = 1 \text{ or } S_{ji} = 1 \right\}$$

The collection *D* can be calculated in parallel using an exchange of weights for the cross-processor influences. The coarse/fine variables are then chosen and the weights updated in the following fashion:

```
for each i ∈ D
        for each j that influences i
                decrement w(j)
                set S_ij = 0
        for each j that depends on i
                for each k that depends on j
                        if k depends on i
                                decrement w(j)
                                set S_kj = 0
```

Whenever, $w(j) < 1$ then j is marked as a fine variable. Finally, mark all the variables in D as coarse variables. Another global exchange of weights and coarse/fine variable selection is made and the algorithm proceeds with the updated influence matrix until all variables are assigned to be either coarse or fine.

The heuristic applied in CLJP coarsening guarantees every fine variable, i, is influenced by at least one coarse variable and moreover if i is influenced by another fine variable, j, then j is influenced by a coarse variable which also influences i. Also, if the same set of global random numbers is used CLJP coarsening will give the identical coarse/fine variable split irrespective of the number of processors.

The PMIS algorithm is simpler as it relaxes the condition that every fine variable that influences a fine variable shares a common influencing coarse variable. Weights are assigned for each variable just as in CLJP, $w(i) = \left|S_i^T\right| + \sigma(i)$ and the collection D retains the same definition. The coarse/fine variables are then chosen and the weights updated in the following fashion:

for each $i \in D$
    mark i as coarse
    for each j that depends on i
        mark j as fine

This is repeated using the updated influence matrix until all variables are marked.

Interpolation algorithm

The coarsening algorithm has split the variable set into coarse, C, and fine, F, variables. As previously stated CLJP coarsening guarantees every fine variable is influenced by at least one coarse variable and an influencing fine variable shares a common influencing coarse variable. This means it is possible to use direct interpolation with separation of weights as the interpolation operator (ref. 14). Put $N_i^+ = \left\{ j : a_{ij} > 0 \right\}$, $N_i^- = \left\{ j : a_{ij} < 0 \right\}$ and $C_i^{S+} = C \cap S_i \cap N_i^+$, $C_i^{S-} = C \cap S_i \cap N_i^-$ where C is the collection of coarse variables. Then for each fine variable i and coarse variable $j \in C_i^s = C \cap S_i$ define:

$$\alpha_i = \frac{\sum_{j \in N_i^-} a_{ij}}{\sum_{j \in C_i^{S-}} a_{ij}} \qquad \text{and} \qquad \beta_i = \frac{\sum_{j \in N_i^+} a_{ij}}{\sum_{j \in C_i^{S+}} a_{ij}}.$$

The interpolation weight for a fine variable i influenced by a coarse variable j, is defined to be:

$$\omega_{ij} = \begin{cases} -\alpha_i a_{ij} / a_{ii} & j \in C_i^{S-} \\ -\beta_i a_{ij} / a_{ii} & j \in C_i^{S+}. \end{cases}$$

Direct interpolation only requires the entries on the given row. If whole rows are stored on each processor then no parallel communication is necessary.

When PMIS coarsening is used, direct interpolation is insufficient and a long range interpolation operator is required. This is chosen to be extended+i interpolation (ref. 18) and is defined as follows. First the coarse variables to interpolate are expanded to include neighbors, let $F_i^s = F \cap S_i$ and put $\hat{C}_i = C_i^S \cup \bigcup_{j \in F_i^S} C_j^S$ which will be the new collection of interpolatory variables. To avoid vanishing denominators in the equations that follow it is necessary to consider the signs of off-diagonal elements. This is done by defining:

$$\overline{a}_{ij} = \begin{cases} 0 & \text{if sign}(a_{ij}) = \text{sign}(a_{ii}), \\ a_{ij} & \text{otherwise.} \end{cases}$$

The interpolation weight for a fine variable i and a coarse variable $j \in \hat{C}_i$ is then defined to be:

$$\omega_{ij} = -\frac{1}{\tilde{a}_{ii}}\left(a_{ij} + \sum_{k\in F_i^S} a_{ik}\frac{\overline{a}_{kj}}{\sum_{l\in\hat{C}_i\cup\{i\}}\overline{a}_{kl}}\right) \text{ where } \tilde{a}_{ii} = \left(a_{ii} + \sum_{m\in N_i\setminus(S_i\cup\hat{C}_i)} a_{im} + \sum_{k\in F_i^S} a_{ik}\frac{\overline{a}_{ki}}{\sum_{l\in\hat{C}_i\cup\{i\}}\overline{a}_{kl}}\right)$$

For both interpolation algorithms the collection of weights defines a rectangular matrix which forms the interpolation operator. The restriction operator is given by taking the transpose of interpolation, which requires some global communication to swap columns and rows.

Relaxation algorithm

The relaxation algorithm used is one iteration of Gauss-Seidel when coarsening and when interpolating. As the variables have been split into coarse and fine variables, it is possible to relax first on the fine variables and then on the coarse variables. This is done both in the restriction and in the interpolation phase of the V-cycle solve. Gauss-Seidel relaxation is inherently sequential, and is parallelized using processor block Gauss-Seidel: this performs Gauss-Seidel on the on-processor values and ignores all off-processor values.

Data Structures

To build a parallel solver requires parallel data structures. This work uses the Compressed Sparse Row (CSR) format as a flexible matrix representation (ref. 19). The CSR representation has three vectors, an integer array, `ptr`, which indexes the start of each row, an integer array, `ind`, which contains the column indices of each non-zero entry, and a float array, `a`, with the actual matrix values. Using this representation it is possible to implement a sparse matrix-vector multiply, *x=Ay* in the following manner:

```
for(i=0;i<nRows;i++)
    x[i]=0;
    for(j=ptr[i];j<ptr[i+1];j++)
            x[i]=x[i]+ a[j]*y[ind[j]];
```

Complete rows are stored on a single processor and matrix elements are split into two parts, an on-processor part for all those indices which are on this processor and an off-processor part for all indices which are off-processor. Assuming a good distribution of rows across processors, this flexible data structure offers good parallel performance for standard matrix operations such as matrix-matrix multiplication, matrix transpose and matrix-vector multiplications.

5. The Coupled Solver for Fully Implicit Cases

First Stage Preconditioning

The first stage of the two stage preconditioner works only on the pressure equation. There are many ways an AMG solver can be used at this stage. For this work the simplest approach is taken and one V-cycle is used.

Second Stage Preconditioning

The second stage of the two stage preconditioner works on all the equations simultaneously. This can be taken to be a fairly weak solver, such as a simultaneous incomplete LU factorization. In this work a forward then backward substitution is used which can be expressed efficiently as the inverse of $(\gamma + L)\gamma^{-1}(\gamma + U)$, where $\gamma = D$ the diagonal of the matrix. The second stage of the preconditioner works on the whole matrix and is parallelized using a block method. On each processor a set of planes of cells is defined in an areal direction (for example a plane may be a set of cells with a given I-index in a Cartesian (I,J,K) indexing scheme). Each processor starts its forward substitution on a plane-by-plane basis. After completing the first plane, the value of which will be needed by the lower processors, the plane is sent using a message passing protocol. The send is non-blocking and can occur whilst later planes are solved for. Before solving for its last planes each processor then has to receive the message sent by the upper processors. The plane-by-plane solver is repeated in reverse for the backward substitution. Effectively this is a two-colour algorithm, with the red planes being all but the last plane on each processor, and the black plane being the last plane. All the red planes are placed first in the solution order, are decoupled and may be solved for independently; after that the red-black coupling terms are treated and then all the black planes solved for independently.

Incorporation of the two-stage preconditioning within GMRES acceleration

As mentioned in section 2, the simple sum of the two corrections, $\begin{pmatrix} x_p \\ 0 \end{pmatrix} + y$, is returned to the GMRES outer iteration. As sub-iterations are not performed (i.e. one iteration of Gauss-Seidel is used as the relaxation operator) a flexible FGMRES (ref. 20) method is not required. Experiments show that performing additional iterations of either the first or second stage preconditioners will reduce the number of overall iterations; but the additional cost of each iteration outweighs the gain and there is no overall benefit in performing sub-iterations.

An alternative non-linear pressure equation

The Quasi-IMPES decoupling strategy described above is not completely justified in that flow terms can be important, and the use of upstreaming in selecting the mobility to be used in a given flow term may imply a strong coupling of the form $\partial R_{ci}/\partial S_{sj}$, where i and j are cell indices. The residual can be viewed as comprising a mass accumulation term M and a flow term F. Depending on the direction of flow, for a given equation the strong derivative $\partial F_{ci}/\partial S_{sj}$ may be on the diagonal or the band. An option is to take the column sum of the matrix – as each flow derivative for flow from cell i to cell j is matched with an equal and opposite one from cell j to cell i this will leave just the mass operator derivatives. An alternative is to attempt to form an approximate pressure equation prior to solving the linear solver equations. In the case in which molar variables are used, a pressure equation may be naturally obtained from the volume balance residual. (In the molar variable case we have (N_c+1) variables per cell, one pressure and N_c molar densities, so there are N_c conservation equations and a volume balance condition that the pore volume and the fluid volume are equal). In the IMPES case each molar variable is eliminated from the volume balance equation to form a pressure equation: this is done using a set of pivot operations. In the fully implicit case it is possible to carry out exactly the same set of pivot operations. These do not form a true pressure equation due to flow term derivatives, except in the limit in which the flows are negligible. This pivoting operation outside the linear solver is thus similar to the Quasi-IMPES diagonal Gaussian elimination described in section 2, except that flow terms are really ignored rather than being treated to the extent to which they fall on the diagonal. Having set up a pressure equation externally, there is no need to perform further pivoting within the linear solver to form a pressure equation for AMG; however the results have been found to be broadly similar to the Quasi-IMPES method of section 2, but with some additional overhead in the extra equation setup. In the following this formulation is called FIPE (Fully Implicit Pressure Equation).

One other option for decoupling would be to aim for a best approach to the perfect Schur complement decoupling mentioned in section 2, possibly using a degree of sparsification. For small problems it is possible to test this method by performing the Schur complement exactly. Surprisingly, this results in a set of pressure equations (which implicitly include the effect of implied saturation changes) which AMG has some difficulty solving (ref. 21).

6. Results

In many cases in which the nested factorisation solver converges well – generally those dominated by a strong vertical transmissibility – it can still be competitive with the AMG method described. The reason is the significantly lower operation count for nested factorization. In fact, the solve stage of nested factorization is only about twice the operation count of an ILU solve (ref. 2).

For the purposes of this paper the second model of the tenth SPE comparison project was used (ref. 22). SPE10 is a challenging model for traditional linear solvers, such as nested factorization, due to the high permeability streaks in the model. It has been demonstrated that the CPR or combinative techniques with an AMG pressure solver can be an effective solver for this problem (refs. 7, 8 and 11). In this work the non-linear solver is converged to a tolerance of 10^{-3}, while the linear solver is converged to a relative tolerance of 10^{-3} when comparing single processor results and 10^{-5} when comparing parallel results.

The results in table 1 compare the AMG solver using CLJP and direct interpolation, with PMIS and extended+i interpolation for the fully-implicit solver on SPE10 for the regular black-oil formation and the FIPE formulation. Percentages are given in terms of the total runtime: for SPE10 the linear solver takes the majority of the total runtime. It is possible to see that FIPE is providing a slightly easier matrix solve. However, the cost in the ILU solver is increased considerably (in absolute terms it is approximately twice as great, each entry in the Jacobian goes from a 2x2 matrix to a 3x3 matrix). PMIS+extended+i is giving similar convergence rates to CLJP+direct, but is considerably faster: this is because PMIS has fewer coarse variables at each level, which means there are fewer levels in the hierarchy and one might expect the matrices to be more dense as a result. However, this does not seem to be the case with extended+i interpolation.

Table 1. Comparison of serial timings for different formulations and coarsening/interpolation algorithms on SPE10

	Quasi-IMPES		FIPE	
	CLJP+direct	PMIS+extended+i	CLJP+direct	PMIS+extended+i
Number of time-steps	128	128	129	129
Number of non-linear iterations	545	542	609	608
Number of linear iterations	11311	7916	7911	7725
Number of linear iterations per non-linear iteration	21	15	13	13
% AMG analyse	19	21	21	18
% ILU analyse	0.5	1	1	1
% AMG solve	38	28	28	22
% ILU solve	9	11	13	16
Time (mins)	523	328	517	424

Table 2 demonstrates scaling results for CLJP and direct interpolation on SPE10. The two-stage solver is showing good parallel scaling, in the sense that the number of linear iterations is not increasing too much as the number of processors is increased. This is typically not the case with the more traditional solvers such as parallel nested factorization (ref. 23).

Table 2. Parallel scaling results for SPE10 using FIPE with CLJP coarsening, direct interpolation and 10^{-5} linear tolerance

	Number of processors				
	1	2	4	8	16
Number of time steps	129	129	129	131	131
Number of non-linear iterations	608	608	608	611	611
Number of linear iterations	12284	12306	12367	12719	13099

The two-stage preconditioner simply adds the results of the two preconditioners. It is possible that a more complex combination of the two preconditioners could give an improved result. This was tested experimentally by using the two preconditioners alternately within an FGMRES (ref. 20). Table 3 demonstrates the results using the regular black-oil formulation with Quasi-IMPES decoupling, PMIS coarsening and extended+i interpolation. As can be seen the number of FGMRES iterations is almost exactly double that of using the two-stage preconditioner, this is because each preconditioner is now one FGMRES cycle. This test would suggest the combination of AMG and ILU are complimentary for SPE10, the AMG preconditioner acting as a long-range solver, the ILU preconditioner as a short range solver. There is no gain in performance to using the two preconditioners alternately as the cost of FGMRES grows quadratically in the number of FGMRES iterations.

Table 3. Comparison of two-stage preconditioning and an alternating preconditioner

	Two-stage	Alternating
Number of time-steps	128	128
Number of non-linear iterations	542	543
Number of linear iterations	7916	16532

7. Discussion and Conclusions

The combination of an AMG pressure solver and a second stage solve applied to all variables has proved effective on problems which have proved intractable using traditional solvers. The second stage solver takes a significant amount of the computational effort, and may not have the desirable asymptotic scaling with problem size possible with AMG. Parallelization of the ILU solver is possible using a multi-color approach, as in this work, but the quality of the preconditioning may be less good for the reordered ILU algorithms.

An alternative would be to treat all the variables, pressures and saturations, with a simultaneous multigrid solver. This raises some problems with inverse weights, and the number of operations required to perform matrix-matrix multiplications on $N_c \times N_c$ submatrices throughout the solver which rises as $O(N_c^3)$. However such simultaneous AMG solvers have been constructed (ref. 11), and do offer the option of obtaining asymptotic scaling with problem size in the fully implicit case. In any event, it is not clear that the saturation variable changes will benefit from the effects of coarse grid corrections and it may be best to allow these to be eliminated at lower levels of the multi-grid solution.

8. Nomenclature

A Matrix to be solved.
b Right hand side of linear equation set.
D Operator extracting the block diagonal of a matrix.
J The Jacobian *∂R/∂X.*

N Number of solution variables per cell.
N_a Number of active cells in model.
P Pressure.
R Residual error for the flow equations.
x Change in solution variable to be determined by solving linear equation set.
y Solution change from second stage of preconditioner.
X Solution vector for the reservoir model.
Q Sub-matrix used in decoupling operator.

c Component index.
i, j Cell indices.
s Saturation index.
ω Interpolation weight.
F Fine variables.
C Coarse variables.

9. Acknowledgements

We would like to thank Taha Taha of Roxar in the preparation of datasets, Alain Dominguez of Intel for help with improving the performance of our code and Gennady Sarkisov for testing. We would also like to thank Roxar for permission to publish this paper.

10. References

1. Meijerink, J.A and Van Der Vorst, H. A., *'An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix'*, Math. Comp., **31**, (1977), pp148-162.
2. Appleyard, J. R., and Cheshire, I. M., *'Nested factorization'*, paper SPE 12264 presented at Proc. 7th SPE Symp. on Reservoir Simulation, 1983.
3. Wallis, J. R.: *'Incomplete Gaussian Elimination as a Preconditioning for Generalized Conjugate Gradient Acceleration',* paper SPE 12265 presented at the SPE 1983 Reservoir Simulation Symposium, San Francisco, CA., 1983.
4. Wallis, J. R., Kendall, R. P. and Little, T. E., *'Constrained residual acceleration of conjugate residual methods'*, paper SPE 13536 presented at SPE Symposium on Numerical Simulation, Dallas, Texas, 1985.
5. Vinsome, P. K. W.: *'ORTHOMIN, an iterative method for solving sparse banded sets of simultaneous equations'*, paper SPE 5749 presented at SPE Symposium on Numerical Simulation, Los Angeles, California, 1976.
6. Saad, Y. and Schultz, M. H.: *'GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems'*, SIAM J. Sci. Stat. Comput. 7 (1986), 856–869.
7. Cao, H., Tchelepi, H.A., Wallis, J. and Yardumian, H., *'Parallel Scalable Unstructured CPR-Type Linear Solver for Reservoir Simulation'*, paper SPE 96809 presented at Proc. 2005 SPE Ann. Tech. Conf. and Exhibition, Dallas, TX, Oct 2005.
8. Durlofsky, L. J., Aziz, K.: *'Advanced Techniques for Reservoir Simulation and Modeling of Nonconventional wells',* Final report to U. S. Dept. of Energy, contract no. DE-AC26-99BC15213 (2004).
9. Lacroix, S., Vassilevski, Y. and Wheeler, M.: *'Decoupling preconditioners in the implicit parallel accurate reservoir simulator (IPARS)',* Numer. Linear Algebra with applications Vol. 8, No. 8, pp. 537-549, 2001
10. Scheichl, R., Masson, R. and Wendebourg, J., *'Decoupling and Block Preconditioning for Sedimentary Basin Simulations'*, Comp. Geosciences 7. pp.295-318, 2003.
11. Stüben, K., Clees, T., Klie, H. and Wheeler, M.: *'Algebraic Multigrid Methods (AMG) for the Efficient Solution of Fully Implicit Formulations in Reservoir Simulation'*, paper SPE 105832 presented at 2007 SPE Reservoir Simulation Symposium, Houston, TX
12. Behie, A. and Vinsome, P.: *'Block iterative methods for fully implicit reservoir simulation'*, paper SPE 9303 presented at 55th Annual Fall Technical Conference, Dallas, September, 1982.
13. Briggs, W. L., Henson V. E. and McCormick, S. F.: *'A Multigrid Tutorial'*, Second Edition. SIAM, 2000.
14. Stüben, K.: *'Algebraic Multigrid'* in Trottenberg, U., Oosterlee, C., Schuller, A.: *'Multigrid'*. Elsevier Academic Press, 2001.
15. Cleary, A. J., Falgout, R. D., Henson, V. E. and Jones, J. E.: *'Coarse grid selection for parallel algebraic multigrid'*, in Proceedings of the fifth international symposium on solving irregularly structured problems in parallel, Spring-Verlag, New York, 1998.
16. Henson, V. E. and Yang, U. M.: *'BoomerAMG: A parallel algebraic multigrid solver and preconditioner'*, Applied Numer. Math 41 (2002), 155–177.
17. De Sterck, H., Yang, U. M. and Heys, J. J.: *'Reducing complexity in parallel algebraic mulitgrid preconditioners'*, SIAM Journal on Matrix Analysis and Applications 27 (2006), 1019–1039.
18. De Sterck, H., Falgout, R. D., Nolting, J. W. and Yang, U. M.: *'Distance-two interpolation for parallel algebraic multigrid'*. Numer. Linear Algebra Appl. 2008; 15:115-139.
19. Barrett, R. et al.: *'Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods',* 2nd Edition, SIAM, 1994.
20. Saad, Y., *'A flexible inner-outer preconditioned GMRES algorithm'*, SIAM Journal on Scientific and Statistical Computing, **14**, 1993.
21. Kwok, W. H. F. K., *'Scalable linear and nonlinear algorithms for multiphase flow in porous media'*. PhD. Thesis, Stanford, 2007.
22. Christie, M.A. and Blunt, M.J.: *'Tenth SPE Comparative Solution Project: A Comparison of Upscaling Techniques'*, paper SPE 72469 in SPE Reservoir Evaluation & Engineering Vol. 4, Num. 4, Aug 2001.
23. Burrows, R., Ponting, D. and Wood, L., *'Parallel Reservoir Simulation with Nested Factorisation'*, paper presented at ECMOR V, 1996.