JOURNAL OF COMPUTATIONAL PHYSICS 59, 290–307 (1985)

Multi-grid Methods for Oil Reservoir Simulation in Two and Three Dimensions

T. SCOTT

Atomic Energy Establishment, Winfrith, Dorchester, Dorset DT2 8DH, England

Received March 7, 1984; revised August 30, 1984

One of the essential components of an oil reservoir simulator is an efficient technique to solve the systems of algebraic equations arising from discretisation of the governing partial differential equations. This paper describes the development of a robust and efficient multi-grid solution algorithm and its implementation for solving typical problems encountered in the modelling of petroleum reservoirs. The method is used to compute results for two- and three-dimensional cases, involving in the region of up to 5000 grid-blocks. A comparison in terms of CRAY CPU time is provided between different multi-grid smoothing strategies employed by the algorithm. © 1985 Academic Press, Inc.

1. INTRODUCTION

The purpose of this paper is to describe the development and application of a selected multi-grid method to solve the nonseparable diffusion equation defined in up to three dimensions. This type of equation plays a central role in the theory of flow through porous media and provides the basis for setting up the governing partial differential equations of oil reservoir simulation. Allowing for discretisation in time, the diffusion equation at each time step may be written compactly as a self-adjoint boundary value problem,

$$\nabla \cdot \mathbf{K} \nabla u = F. \tag{1.1}$$

Concerning notation, $\mathbf{K}$ is proportional to the permeability tensor, u is the dependent variable, in this case pressure, and F a functional term evaluated at the previous time step. The problem is to solve for u implicitly throughout the reservoir domain Ω subject to prescribed boundary conditions (usually no-flow Neumann conditions) on the boundary $\partial\Omega$.

The approach to deriving a solution to Eq. (1.1) is to discretise the system using finite differences. Hence the continuous problem is reduced to a discrete system of linear equations to be solved for u. It is common practice in oil reservoir simulation to adopt a block centred grid structure whereby the region Ω is divided up into blocks and the pressure solved for at the centre of each block.

Given this block centred scheme, the objective has been to develop and apply an

0021-9991/85 $3.00

efficient and robust multi-grid algorithm to solve the approximating difference equations. By robust is meant a technique which can be successfully applied to a wide variety of domains with properties defined according to the second order tensor **K**. Thus **K** may involve sharp discontinuities due, for example, to faults in the rock matrix or exhibit large anisotropic variations from one region of the reservoir to another.

Hence the task required the identification of a reliable multi-grid algorithm which could quickly and efficiently solve upwards of 700 linear equations involving the pressure distribution. Further it was necessary to derive a method for obtaining the coarse-grid difference equations from the previous fine-grid difference equations since only the coefficient matrix is readily available rather than the partial differential equation itself. Earlier multi-grid work [4] relied on knowledge of the differential equation itself being at hand so that coarse-grid representations of the finite difference equations could be formulated by returning to and discretising this equation as necessary.

In response to these demands a multi-grid solution package has been developed which can be added to an existing oil reservoir simulator to provide an alternative iterative solution procedure. This solution package is particularly suitable for field studies involving up to several thousand grid-blocks.

Similar work has been carried out independently by Behie and Forsyth [3] in connection with oil reservoir simulation. In the present work, however, interpolation and the number of multi-grid relaxation sweeps performed at each level of coarseness is discussed more generally. More significant is the fact that use is made of a full multi-grid algorithm to derive the results reported in this paper rather than the cyclic approach adopted in [3]. Comparison between applications to two- and three-dimensional examples is restricted in the present study to techniques using similar mathematical operators to measure directly the relative success of such techniques for use in solving these problems. The work in [3] is mainly concerned with comparing multi-grid incorporating plane relaxation and the method of incomplete decomposition combined with conjugate gradients for three-dimensional problems alone. Moreover, in the current study comparisons are made in terms of CPU time on the CRAY-1 rather than numbers of arithmetic operations.

Section 2 begins with a brief description of the multi-grid strategy used in the present work, before going on to discuss the finer mathematical points and details of the computational aspects. Application and results of the method to representative examples form the material of Section 3 and concluding remarks are presented in Section 4.

2. Multi-Grid Algorithm

To establish notation and for completeness, a brief description is first given of the multi-grid strategy used to solve Eq. (1.1). It is assumed throughout that the time dependence of the diffusion equation has been discretised and that one is now con-

cerned with an elliptic boundary value problem to be solved at each time step. Rectangular Cartesian coordinates are used as a basis for spatial discretisation. The chief objective is to set up a particular multi-grid algorithm and to then use it to solve representative examples in two and three dimensions to determine the relative success of the constituent operators used for such items as relaxation and interpolation.

The algorithm is written with the option of starting iteration with either some user-supplied initial approximation to the solution on the finest level (as in [3]) or alternatively at the coarsest level of discretisation with a direct solution method. Thus the algorithm incorporates a cyclic option or the full multi-grid procedure [6]. Both work in a fixed mode, determined after a number of trial runs on representative example problems. The so-called automatic prescription developed by Nicolaides [8] within the context of general finite element systems is employed to compute the various grid communication operators and coarse level difference equations.

2.1. *Brief Outline*

Let $G^1,\ldots, G^M$ denote a sequence of discretisation grids such that $G^k \subset G^{k+1}$ with G^k a typical member of this grid family. G^M represents the finest grid and G^1 the coarsest. The centres of the grid-blocks referred to in Section 1 are located at the nodes of G^k. Suppose the corresponding mesh sizes are $h_1 > \cdots > h_M$ where for simplicity a square uniform grid is assumed; standard coarsening is used to define the step-lengths, namely $h_k = 2h_{k+1}$.

The discretised form of Eq. (1.1) is expressed as

$$L^M U^M = F^M \tag{2.1}$$

on G^M, the grid on which the problem is presented to the multi-grid package. L^M represents the finite difference form of the differential operator appearing in Eq. (1.1) and U^M the exact solution to this system of linear equations. It is assumed that iteration is to begin on G^M with some initial approximation u_0^M to U^M; the extension to full multi-grid iteration is relatively straightforward.

The starting point is to write Eq. (2.1) in residual form on G^M, that is,

$$L^M v^M = f^M \tag{2.2}$$

where $v^M = U^M - u^M$ and f^M is the residual function; u^M is some approximate solution to U^M so that v^M is interpreted as a correction term. As is well known from the multi-grid literature, on a typical grid G^k, it is the correction v^k which is smoothed by relaxation on the (coarser) auxiliary grids comprising the grid hierarchy. That is, those error components with Fourier wavelength of the order of h_k are most efficiently eliminated by just one or two relaxation sweeps on G^k.

Having smoothed v^k in this way, the problem is then transferred to G^{k-1} according to the technique

$$L^{k-1} v^{k-1} = f^{k-1}, \tag{2.3}$$

where the residual term f^{k-1} is defined to be given by

$$f^{k-1} = I_k^{k-1}(f^k - L^k v^k) \tag{2.4}$$

with I_k^{k-1} a suitable residual transfer operator. The usual smoothing procedure is again repeated, this time on G^{k-1}, whereafter v^{k-1} may be interpreted as a coarse-grid correction to v^k. The initial approximation required to begin relaxation on Eq. (2.3) is taken to be the null solution. Having solved the problem on G^{k-1} sufficiently well, the function v^k is corrected as

$$v^k \to v^k + I_{k-1}^k v^{k-1} \tag{2.5}$$

where I_{k-1}^k denotes interpolation from G^{k-1} to G^k. The choice of interpolation and residual transfer operator is discussed in the following section. Hence the correction v^k has been modified according to line (2.5) and should be a better value for the next step in the algorithm. This new value is further smoothed by relaxation before being used as a correction to v^{k+1} on the next finer grid G^{k+1}. On the other hand if $k = M$, this value is treated as the new approximate solution to the problem on G^M (Eq. (2.1)) and checked for convergence.

The above sequence of steps is repeated for all $k \leqslant M$, cycling down to the coarsest level G^1 and returning to the finest level G^M. Convergence is investigated by comparing the L_2 norm of the residual with some prescribed tolerance ε, usually $\varepsilon = 0.01$. In the current program of work, the coarsest grid equations are solved directly by Gaussian elimination rather than by relaxation. This is relatively cheap compared to calculations on other grids due to the reduction in the number of grid points at this level of discretisation.

The algorithm described above is essentially similar to the Correction Scheme introduced by Brandt [4]. Computations begin on G^M by setting $v^M = u_0^M$ rather than zero and applying relaxation before transferring the problem to G^{M-1}. Moreover, it is essential to this scheme that the partial differential equation be linear in order that a meaningful equation can be written down for v^k.

Alternatively u_0^M may be calculated by the package by starting on G^1 with a direct solution method and interpolating to G^M by a process of multi-grid cycling. This relieves the user of having to supply a suitable u_0^M and is particularly useful at the first time step where a good initial starting value is not always available. Hence full multi-grid iteration is applied at the first time step. At subsequent time steps, because the pressure distribution is generally a slowly varying function within the overall solution strategy, the previous distribution at the last time step can be used as an initial starting approximation on the finest grid for the new time step. This is referred to as a purely cyclic algorithm.

The multi-grid strategy outlined above defines the basic framework of the current algorithm developed for simulation studies. Some further details are covered in the next section.

2.2. *Mathematical Operators*

Having established the principal features of the multi-grid algorithm, it is now appropriate to specify in more detail the choice of the various mathematical operators constituting the scheme, such as those concerned with communication between grids and smoothing.

As already mentioned in Section 2.1, at the first time step the basic procedure is the full multi-grid approach whereby in order to obtain a good initial approximate solution u_0^M, the algorithm begins on G^1 where the system of linear equations is solved exactly by a non-iterative scheme. This solution is interpolated to G^2, refined by a multi-grid cycle and then interpolated to G^3. This is repeated until the algorithm reaches G^M and hence provides a good starting solution at this level. Ideally only a few multi-grid sweeps on this approximation are necessary to yield an acceptable solution to Eq. (2.1). It is important to note that this technique eliminates the extra work which would be incurred by an unfortunate starting approximation on G^M in the cyclic multi-grid approach [4].

In addition, a fixed mode of operation is chosen as opposed to the accommodative mode. That is, instead of inserting switching criteria to determine how many relaxation sweeps need to be performed on a given grid G^k $(k > 1)$ before the algorithm should switch to another grid, the number of sweeps per grid is prescribed in advance. In other words the number is fixed at the outset, taking into account the nature of the reservoir problem and the effectiveness of the selected relaxation procedure at smoothing the error correction term. Moreover this mode of operation thus eliminates the additional code required to implement switching criteria and hence a saving on execution time.

The mathematical operators are described below in turn.

Relaxation. Three options have been considered:

(i) point successive relaxation,

(ii) line successive relaxation,

(iii) alternating line relaxation.

The first is straightforward application of Gauss–Seidel iteration where the solution at each grid point is updated in turn using the latest values available at neighbouring points. Standard ordering is adopted.

The second option updates a line of point values simultaneously, the direction of the line being parallel to either one of the three co-ordinate axes (x, y, or z). This is particularly useful for solving problems involving a discontinuity in the coefficient terms $\mathbf{K}$ of Eq. (1.1) where the line of relaxation should be optimised with respect to the line of discontinuity.

The third choice combines line relaxation of option (ii) in such a way that in three dimensions a single sweep is defined to incorporate taking lines parallel to first the x axis, then the y axis, and third, the z axis. For problems which exhibit

TABLE I

CRAY CPU Time (in seconds) Required to Solve Two-Dimensional Examples by Multi-Grid Iteration[a]

Example	GD4	PSR	LSR(x)	LSR(y)	LSR(x, y)
			33×33 Grid-blocks		
1	0.231	—	—	0.075 (4)	0.110 (4)
2	0.231	0.089 (7)	0.090 (6)	0.099 (6)	0.128 (5)
3	0.231	—	—	0.099 (6)	0.128 (5)
			65×65 Grid-blocks		
1	2.770	—	—	0.322 (6)	0.502 (6)
2	2.773	0.337 (9)	0.392 (9)	0.439 (9)	0.502 (6)
3	2.772	—	—	0.440 (9)	0.566 (7)

[a] Number of multi-grid V-cycles given in parentheses.

TABLE II

CRAY CPU Time (in seconds) Required to Solve Three-Dimensional Examples by Multi-Grid Iteration[a]

Example	GD4	PSR	LSR(x)	LSR(y)	LSR(z)	LSR(x, y, z)
			$9 \times 9 \times 9$ Grid-blocks			
1	0.573	—	—	0.083 (4)	—	0.126 (3)
2	0.573	0.082 (5)	0.076 (4)	0.083 (4)	0.084 (4)	0.101 (2)
3	0.573	—	—	—	0.084 (4)	0.126 (3)
			$17 \times 17 \times 17$ Grid-blocks			
1	—	—	—	0.511 (5)	—	0.896 (4)
2	—	0.475 (6)	0.512 (6)	0.575 (6)	0.517 (5)	0.746 (3)
3	—	—	—	—	0.517 (5)	0.896 (4)

[a] Number of multi-grid V-cycles given in parentheses.

TABLE III

Average CRAY CPU Time (in seconds) Required for Multi-Grid Preliminary Calculations (MGPC) and for Multi-Grid V-Cycles using Point Successive Relaxation MGCY(PSR), Line Successive Relaxation MGCY(LSR), and Alternating Line Relaxation MGCY(ALR)

	Two-dimensional examples	
	33×33 Grid	65×65 Grid
MGPC	0.010	0.036
MGCY(PSR)	0.011	0.035
MGCY(LSR)	0.014	0.046
MGCY(ALR)	0.023	0.082
	Three-dimensional examples	
	$9 \times 9 \times 9$ Grid	$17 \times 17 \times 17$ Grid
MGPC	0.022	0.118
MGCY(PSR)	0.011	0.061
MGCY(LSR)	0.013	0.072
MGCY(ALR)	0.031	0.193

severe anisotropy, alternating line relaxation is particularly appropriate. It can also be used to treat efficiently the difficulties associated with discontinuities.

None of the above relaxation procedures requires an iteration parameter as, for example, in the more conventional method of line successive overrelaxation. Each of these methods has been consistently applied to problems in two and three dimensions and a comparison drawn up in Tables I–III.

Interpolation. This is concerned with transfer of the solution vector from G^{k-1} to G^k and in Section 2.1 is denoted by the symbol I^k_{k-1}. For fine-grid points which coincide with coarse-grid points, the identity operator is applied. Otherwise use is made of the discretisation equation about the particular fine-grid point labelled (i, j, k) for which one wants to interpolate a value

$$L^q_{i,j,k} u^q_{i,j,k} = f^q_{i,j,k}, \tag{2.6}$$

where $1 < q < M$. The inclusion of the right-hand side, $f^q_{i,j,k}$, is optional. In the derivations to be described in this section it is, however, ignored to retain a higher degree of clarity. The multi-grid results reported in Tables I–III have been obtained using an interpolation operator which excludes $f^q_{i,j,k}$. Generally exclusion of this right-hand-side function was found to lead to a more efficient interpolation scheme. Referring to Fig. 1 there are in three dimensions 3 types of points to which Eq. (2.6) is applied.

The first type consists of those points which lie on coarse-grid lines but do not coincide with coarse-grid points. For such points Eq. (2.6) is averaged in the two

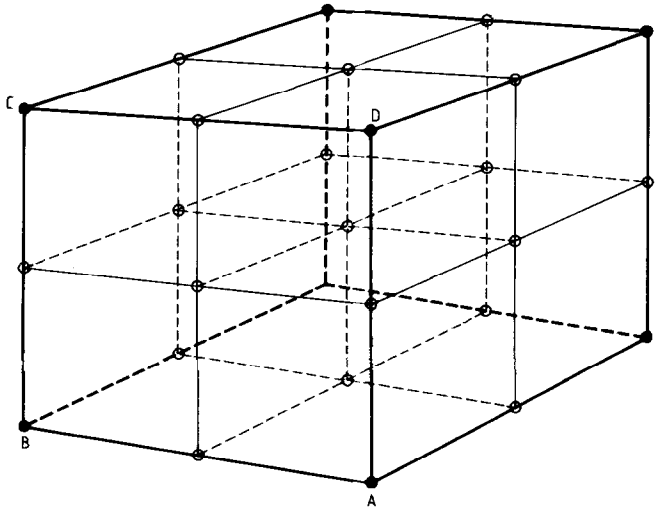

FIG. 1. Diagram to illustrate 3-dimensional interpolation. (○) Fine-grid points; (●) coarse-grid points. *AB* is an example of a coarse-grid line; *ABCD* is an example of a coarse-grid plane.

directions perpendicular to the coarse-grid line in question (one direction, if the problem is two dimensional). This simply amounts to adding the coefficients of $L^q_{i,j,k}$ in the respective direction(s). The result is an equation in the three fine-grid points which lie on the coarse-grid line. The two end-points coincide with coarse-grid points (by definition of the grid structure) and are therefore known. On substitution for these points, the equation is then solved for the unknown (mid-point) value.

The second type of point consists of those on coarse-grid planes but not lying on coarse-grid lines. In two dimensions, using the above techniques to obtain values at the fine-grid points within the plane which coincide with coarse-grid points and coarse-grid lines, Eq. (2.6) is solved for the fine-grid value in question. For three-dimensional cases, Eq. (2.6) is averaged, employing a similar technique to that described above, in the direction perpendicular to the plane. The resulting equation (defined in a plane) is then treated as if it were a two-dimensional example.

The final type of point only occurs in three-dimensional problems and does not lie on a coarse-grid plane (rather the centre point of the cube illustrated in Fig. 1). However, the 26 neighbouring (nearest) fine-grid point values are all known from application of the above averaging processes (8 points coincide with coarse-grid points, 12 points lie at the mid-point of coarse-grid lines, 6 points lie at the centre of coarse-grid planes) and hence no averaging is needed. Equation (2.6) is solved for the centre point value using the 26 previously interpolated values.

It is noted that because the central difference Eq. (2.6) is used to derive the interpolation formulae, maximum use is made of the information supplied by the discrete problem. As a further note, this approach to interpolation preserves continuity

of $\mathbf{K}\nabla u$ (Eq. (1.1)) which is more accurate physically than ordinary linear interpolation which only preserves continuity of ∇u. The scheme adopted here is more appropriate to petroleum reservoir calculations which frequently involve jump discontinuities of several orders of magnitude in $\mathbf{K}$. Ordinary linear interpolation is only reasonably accurate for problems involving changes in $\mathbf{K}$ of no more than a single order of magnitude.

Residual Transfer. This refers to the definition of the operator I_k^{k-1} appearing in Eq. (2.4) which is used to transfer the residual from G^k to G^{k-1}. Preliminary applications of multi-grid merely used the fine-grid values which happened to coincide with the required coarse-grid points. This technique, sometimes referred to as injection, has, however, been shown by Brandt [5] to lead to a 40% degradation in the rate of convergence for Neumann problems compared to the corresponding Dirichlet problem. Such difficulty is overcome by transferring weighted averages of neighbouring residuals instead. Indeed if L^k possesses highly varying coefficients, the residuals after relaxation are also highly varying so that it becomes necessary to use some residual weighting scheme to correctly represent them on G^{k-1}. The approach adopted in the present study follows along these lines and is more robust and consistent with the interpolation process.

In fact the residual transfer operator is defined to be the transpose of the interpolation operator. That is, in three dimensions, for example, one writes down the interpolation formula for the third type of point discussed in the above section on interpolation. Each of the 26 neighbouring point values are weighted in some way, namely the interpolation weights. These weight coefficients are transposed with respect to opposite points in the discretisation molecule so that the residual at the coarse-grid point in question is obtained as a weighted sum of the residuals at the 26 nearest fine-grid points.

Coarse-Grid Difference Operator. Given the fine-grid difference operator L^k, the interpolation operator (I_{k-1}^k) and hence the residual transfer operator $(I_{k-1}^k)^T$, then following Nicolaides [8], an automatic way to construct the coarse-grid operator is by the prescription

$$L^{k-1} = (I_{k-1}^k)^T L^k I_{k-1}^k. \tag{2.7}$$

The action of the product of operators on the right-hand side can be elucidated by considering its effect on some coarse-grid function $u_{I,J,K}$. Upper case subscripts are employed to denote coarse-grid values and, in what follows, lower case to denote fine-grid values. The interpolation operator (I_{k-1}^k) transfers from the coarse grid G^{k-1} to the fine grid G^k whence the fine-grid difference operator L^k acts on each of the $u_{i,j,k}$ variables. The residual transfer (or restriction) operator $(I_{k-1}^k)^T$ then returns the problem to the coarse grid, thus providing the coefficients of each of the $u_{I,J,K}$ comprising the discretisation molecule on G^{k-1}.

The derivation of the coarse-grid difference operator coefficients is a tedious task

and involves lengthy algebraic formalism. Even if L^k is based on a 7-point discretisation molecule (5-point in two dimensions), L^{k-1} will involve 27 points (9 in two dimensions). On subsequent coarser grids the L^{k-1} discretisation molecule will again comprise 27 points in three dimensions, 9 in two dimensions. The final form of the coarse-grid difference operators is a long and complicated expression, particularly in three dimensions. Various cross-checks on the algebra are maintained by taking advantage of the symmetry of the problem. A similar approach to the coarse-grid difference operator, summarised by Eq. (2.7), has been utilised by Alcouffe *et al.* [1] and Dendy [7] who both confined their studies to two-dimensional problems and also by Behie and Forsyth [3].

2.3. *Computational Details*

Having outlined the mathematical operators to be employed, consideration is now given to some of the other details of the multi-grid algorithm. The grid

boundary of the problem is included on each grid, a fictitious set of boundary points is added outside the domain of interest, the distance of these points from the true boundary depending on the step-length of the grid to which the points belong. This means that the boundary of each coarse grid is not identical with the boundary of any other grid. Essentially an aid to programming [7], the foregoing may be summarised by saying that if a typical grid point is denoted by (i, j, k), then the grid itself is defined to be the set $\{(i, j, k)\colon 1 \leqslant i \leqslant i_m,\ 1 \leqslant j \leqslant j_m,\ 1 \leqslant k \leqslant k_m\}$ with i_m, j_m, k_m the maximum number of points in the x, y, and z directions, respectively. The fictitious boundary points are given by the sets,

$$\{(1, j, k),\ (i_m, j, k)\colon 1 \leqslant j \leqslant j_m,\ 1 \leqslant k \leqslant k_m\}$$

$$\{(i, 1, k),\ (i, j_m, k)\colon 1 \leqslant i \leqslant i_m,\ 1 \leqslant k \leqslant k_m\}$$

$$\{(i, j, 1),\ (i, j, k_m)\colon 1 \leqslant i \leqslant i_m,\ 1 \leqslant j \leqslant j_m\}.$$

As remarked earlier, on the coarsest level of discretisation the resultant system of difference equations is solved exactly by a direct (Gauss elimination) method. Because the coefficient matrix is unchanged, the upper and lower triangular matrices of the decomposition are stored for solution at this level during subsequent multi-grid cycles. It is recognised that relaxation could be used relatively inexpensively at this stage but solving directly eliminates the additional work necessary if the first approximation used to start the iteration is considerably in error.

The number of relaxation sweeps to be performed on each grid apart from the coarsest is fixed to be one before transfer to another grid, whether up or down the grid hierarchy. The only other exception occurs when the algorithm reaches the finest grid whereupon two sweeps are carried out. Other combinations of relaxation sweeps have been tried such as one sweep when the previous grid was lower down

the hierarchy (a coarser grid), two sweeps when the previous grid was higher up in the hierarchy (a finer grid) and three sweeps at the finest level. However, the first mentioned strategy was found to be generally the most efficient in terms of computational time for the examples discussed in Section 3. Depending on whether or not the prescribed tolerance criterion for solution to the problem has been satisfied at the finest level, the algorithm will either terminate or begin another multi-grid cycle, cycling through all the coarser levels to the coarsest G^1 and returning through the same sequence in reverse order to G^M. Hence the algorithm utilises the so-called V-cycle. Moreover, the algorithm will only move to the grid immediately finer or coarser than the current grid; it will never skip a grid.

On the issue of storage it is necessary to store for each grid G^k, $k = 1,..., M-1$, the interpolation weights and the coefficients of the difference operator L^k. For two-dimensional problems, assuming a 5-point differencing scheme for L^M, this amounts to approximately $10n$ locations, where n is the number of unknowns on G^M. In addition, storage has been included for the solution itself and the right-hand side for each of the computational grids. On the other hand for three-dimensional cases with a 7-point discretisation molecule on the finest grid, approximately $12n$ locations are required where again n is the number of unknowns on the finest grid. To conserve on the number of arithmetic operations, additional storage is utilised to retain a number of auxiliary arrays. This increases the allocation to approximately $14n$ and $17n$ locations, respectively, for two and three dimensions. It is perhaps worth saying at this point that a significant proportion of the solution time may be spent in computing the interpolation weights and especially the L^k coefficients. This is found to be particularly true for three-dimensional cases and will be reflected in the results recorded in Tables I–III.

The multi-grid algorithm described in this section has been applied to a number of example problems, selected to include the effects of severe anisotropy and sharp discontinuity in the coefficient terms of **K**, typical of those encountered in petroleum reservoir simulation studies. It is these test problems that are discussed in the following section.

3. Representative Examples

In reservoir simulation, one is frequently concerned with solving a partial differential equation which describes a time evolution process. As already stated in Section 1, the usual strategy is to consider the problem at successive time steps so that it becomes necessary to solve a boundary value problem at each step. The current implementation of the multi-grid package is within a simulator which assumes incompressible flow. Various representative examples have been considered in two and three dimensions which involve anisotropic and discontinuous heterogeneities and allow for injector and producer wells via a relatively simple model.

Returning to the general Eq. (1.1), this is expressed in rectangular Cartesian co-ordinates in the form

$$\frac{\partial}{\partial x} K_x \frac{\partial u}{\partial x} + \frac{\partial}{\partial y} K_y \frac{\partial u}{\partial y} + \frac{\partial}{\partial z} K_z \frac{\partial u}{\partial z} = F, \tag{3.1}$$

where $F = F(x, y, z)$ and $K_\alpha = K_\alpha(x, y, z)$. The analogous equation in two dimensions is defined in the x–y plane. In writing down Eq. (3.1), the common assumption is made that the co-ordinate axes of the reference system are aligned along the principal axes of the tensor $\mathbf{K}$. Different functional forms are considered for (K_x, K_y, K_z) and the right-hand-side function F.

Applying second-order central differences to reduce Eq. (3.1) to a discrete system on the finest grid G^M, one readily obtains a matrix system of linear equations equivalent to Eq. (2.1). In the notation of oil reservoir simulation, the transmissibility terms are defined to be the K_α. The right-hand-side function provides the possibility of including source and sink terms at appropriate points of interest.

The finite difference equations associated with each representative example are solved on the CRAY-1 and a comparison provided in Tables I–III of the times taken with different smoothing operators; vectorised code is employed as much as possible. The time required for solution by Gaussian elimination with $D4$ ordering [9] is also included. This latter solution method is a variant of Gaussian elimination particularly suited to handling the systems of linear algebraic equations arising from discretisation of equations such as Eq. (3.1). The key idea is to order the algebraic equations so that the coefficient matrix assumes a special structure. By performing forward elimination on the lower half of the coefficient matrix ordered in this fashion it is possible to decouple the unknowns, each set having half the number of elements of the original set. In this way, one is only required to solve a matrix equation of just half the dimension of the original matrix equation and to derive the remaining unknowns by a simple back substitution. Further details on the scheme can be found in Refs. [2, 9].

Gaussian elimination with $D4$ ordering represents one of the most widely used direct methods of solution and for this reason is chosen for comparison with the multi-grid method. Use of the scheme is, however, governed to a large extent by the bandwidth of the overall coefficient matrix, so that for larger dimensional matrices the amount of computer storage required may prohibit use of the technique. One could also compare with other iterative methods apart from multi-grid. However, these are generally sensitive to the initial approximate solution and may converge quickly if this choice is close to the exact solution or, on the other hand, grind away for some time if the approximation differs considerably from the exact result. Full multi-grid iteration as used in the present study, however, determines its own good initial starting solution right at the beginning by solving the reduced system of equations at the coarsest level and interpolating this solution to the finest grid using multi-grid cycling. Iteration is terminated when the L_2 norm of the residuals on G^M becomes less than ε, where $\varepsilon = 0.01$.

It is important to note that in this application of multi-grid, the full multi-grid algorithm is employed to derive the results obtained in Tables I–III. This is in contrast to the work of Behie and Forsyth [3] which starts with an initial guess of zero on the finest grid rather than a direct solution approach at the coarsest level. Moreover, it should be noted that in the present work, specific CPU times are tabulated for a given machine which provides for a more readily understood comparison with other methods, than by counting the number of arithmetic operations.

The domain of the representative examples is taken to be square in two dimensions and cubic in three dimensions; the code, however, is not restricted to these configurations and can be readily applied to rectangular regions with any number of grid points along each coordinate axis. Both types of domain are divided up into block-centred grids (a single layer of grid-blocks in two dimensions) with u to be calculated at the centre of each block. **K** denotes the transmissibility between adjacent grid-blocks. The examples have been selected on the basis of variation in **K** and three cases studied on each of the two domains. The two-dimensional examples are discussed first.

3.1. *Two-Dimensional Examples*

As remarked earlier the domain is set in the x–y plane; both 33×33 grid-block ($N = 33$) and 65×65 grid-block ($N = 65$) configurations are considered in turn to define the finest level of discretisation. The step-length is taken to be scaled to unity in both co-ordinate directions at this level. No-flow (Neumann) conditions are imposed along all boundaries. A source with strength proportional to N is located in the corner grid-block labelled (1, 1) and a sink of similar strength in the opposite corner (N, N). Within the context of reservoir engineering these correspond to a simple description of injection and production wells, respectively.

The degree of continuity and isotropy of the medium is governed by **K**. Below are listed the properties of **K** for each of the three problems.

EXAMPLE 1. Continuous and anisotropic everywhere with $K_x = 1$ and $K_y = 10^2$ for $N = 33$ and $N = 65$.

EXAMPLE 2. Discontinuous and isotropic such that for

$$i_1 \leqslant i \leqslant i_2 \text{ and } j_1 \leqslant j \leqslant j_2, \qquad K_x = K_y = 10^{-3}$$

and elsewhere $K_x = K_y = 1$.

$$\text{For } N = 33: \qquad i_1 = j_1 = 12 \text{ and } i_2 = j_2 = 22.$$

$$\text{For } N = 65: \qquad i_1 = j_1 = 22 \text{ and } i_2 = j_2 = 44.$$

EXAMPLE 3. Discontinuous and anisotropic such that $K_x = 1$ everywhere and $K_y = 1$ except for $j_1 \leqslant j \leqslant j_2$ where $K_y = 10^2$ and for $j_3 \leqslant j \leqslant j_4$ where $K_y = 10$.

For $N = 33$: $j_1 = 16, j_2 = 21, j_3 = 22, j_4 = 27.$

For $N = 65$: $j_1 = 30, j_2 = 41, j_3 = 42, j_4 = 53.$

Using multi-grid, the number of auxiliary coarser meshes set up in addition to the finest grid was 4 for the examples with $N = 33$ and 5 for those with $N = 65$.

All problems were solved on a CRAY-1 machine. The amount of CPU time in seconds needed to obtain a convergent solution is recorded in Table I. Apart from the left-hand column headed GD4, which lists the time required to solve by Gaussian elimination preceded by $D4$ ordering, the table provides a summary of the times taken when each of the smoothing procedures discussed in Section 2.2 is employed within the multi-grid scheme. Hence the column under PSR denotes results obtained using point successive relaxation; LSR(α) corresponds to use of line successive relaxation by lines parallel to the direction(s) indicated by α. Results in the column furthest to the right were derived from using alternating line successive relaxation. For each multi-grid run, the time needed to perform the preliminary calculations to set up the various interpolation weights and coarse-grid difference operators is also included in the figure given. This set-up time is also tabulated separately in Table III together with the time required to perform a complete multi-grid V-cycle using the respective smoothing operator. The number of complete multi-grid cycles used by the algorithm to determine the solution is quoted in parentheses below each of the entries in Table I. A dash indicates that an excess of multi-grid cycles (greater than 10) was needed to achieve convergence.

From Table I it is immediately observed that in each case, solution of these problems using multi-grid iteration is faster than using the non-iterative approach. Indeed for such problems which involve 1089 and 4225 unknowns at the finest level of discretisation, one would expect that an iterative method should prove superior over a direct elimination procedure. This is true not only in terms of computational time but also concerning storage demanded by the algorithm; the amount of store needed by the Gauss $D4$ algorithm becomes prohibitively excessive for larger domain problems due to the increased size of the bandwidth.

The speed of convergence within the multi-grid scheme depends crucially on the choice of relaxation. In the case of anisotropic problems it is clearly desirable to choose the optimum direction for relaxation, namely the direction in which K_α is greatest. A similar observation is well known when using line successive over-relaxation (LSOR) to solve the finite difference equations at the finest level of discretisation alone, [2]. On the other hand, Example 2 which includes discontinuity but no anisotropy converges quickly for each choice of relaxation, a result which could be anticipated from the symmetry of the problem.

Generally, given the optimum relaxation method, it is seen from Table I that less than 0.1 s are needed to solve examples on a 33×33 grid and 0.3–0.4 s to solve problems on a 65×65 grid configuration. Use of alternating line relaxation, though requiring fewer relaxation sweeps, is, however, not to be recommended for these examples since for each comparison, more time was necessary to achieve similar

accuracy to using single line relaxation or indeed point successive relaxation in some cases. On the other hand, if time is not of prime importance, smoothing by alternating line relaxation is noted to produce a convergent solution in all cases.

3.2. *Three-Dimensional Examples*

As above in Section 3.1 two levels of finest discretisation are considered, the first comprising $9 \times 9 \times 9$ grid-blocks ($N = 9$) and the second $17 \times 17 \times 17$ grid-blocks ($N = 17$), each grid-block being scaled so as to have edges of length unity. No-flow (Neumann) conditions are imposed over all boundaries of the domain. Source and sink terms consist of a source in the grid-blocks labelled $(1, 1, k)$ with strength proportional to N^2 and a sink diagonally opposite in the grid-blocks with index co-ordinates (N, N, k) of similar strength, $k = 1, 2, \ldots, N$. In oil reservoir terminology, taking the z axis to be in the vertical direction, this corresponds to simple modelling of an injection and a production well completed in all layers of the reservoir.

These examples are essentially three-dimensional geometric analogues of the two-dimensional problems.

EXAMPLE 1. Continuous and anisotropic everywhere with $K_x = 1$, $K_y = 10$ and $K_z = 10^{-1}$ for $N = 9$ and $N = 17$.

EXAMPLE 2. Discontinuous and isotropic such that for

$$i_1 \leqslant i \leqslant i_2,\ j_1 \leqslant j \leqslant j_2, \text{ and } k_1 \leqslant k \leqslant k_2, \qquad K_x = K_y = K_z = 10^{-1}$$

and elsewhere $K_x = K_y = K_z = 1$.

For $N = 9$: $\quad i_1 = j_1 = k_1 = 4$ and $i_2 = j_2 = k_2 = 6$.

For $N = 17$: $\quad i_1 = j_1 = k_1 = 7$ and $i_2 = j_2 = k_2 = 11$.

EXAMPLE 3. Discontinuous and anisotropic such that $K_x = K_y = 1$ everywhere and $K_z = 1$ except for $k_1 \leqslant k \leqslant k_2$ where $K_z = 10^2$ and for $k_3 \leqslant k \leqslant k_4$ where $K_z = 10$.

For $N = 9$: $\quad k_1 = 5,\ k_2 = 5,\ \ k_3 = 6,\ \ k_4 = 7$.

For $N = 17$: $\quad k_1 = 9,\ k_2 = 11,\ k_3 = 12,\ k_4 = 14$.

Apart from the finest grid on which the problems are discretised by the simulator, multi-grid solution of these examples used 2 additional coarser grids when $N = 9$ and 3 additional grids when $N = 17$.

Again all three-dimensional examples were solved on a CRAY-1 machine. The associated CPU times are tabulated in Table II where the notation follows that adopted in Table I. A full description of the symbols and headings has already been given in Section 3.1. The times needed to perform the preliminary calculations for setting up the grid transfer operators and the coarse-grid difference equations are presented in Table III. Also included in this table are the respective times for perfor-

ming complete multi-grid V-cycles starting at the finest grid and cycling down grid by grid to the coarsest level before returning similarly to the finest level. In the case of Gaussian elimination with $D4$ ordering, the routine employed in this study was unable to handle the greater bandwidth encountered on the larger $(17 \times 17 \times 17)$ grid configuration. Otherwise a dash in the columns referring to multi-grid calculations indicates that more than 10 multi-grid iterations (complete cycles) were required to satisfy the convergence criterion.

Inspection of results in Table II reveals a similar behaviour to those reported for the two-dimensional runs in Table I. It is anisotropy which causes the greater difficulty compared to discontinuity; Example 2 which involves only discontinuity is readily amenable to solution using any of the smoothing procedures considered. On the other hand a degree of sensitivity is demonstrated by Examples 1 and 3, which both include anisotropic effects, to the choice of relaxation procedure. Again for these problems, the line of relaxation should be taken in the direction of greatest K_α to achieve maximum efficiency.

On comparing with the two-dimensional results, the increased bandwidth of the present coefficient matrices causes a degradation in execution time as far as the Gauss $D4$ method is concerned. Multi-grid, however, is able to solve the examples on the $9 \times 9 \times 9$ grid (729 points) in a comparatively short time. Indeed, for grids involving in the region of 700 unknowns in the three-dimensional geometry of this study, multi-grid is able to derive a convergent solution in less than 0.1 s. Turning to the larger grid, possessing 4913 unknowns, multi-grid iteration is able to solve the problems in less than 0.5 sec using the optimum relaxation technique. Solution of the three-dimensional problem is generally more time consuming than the two-dimensional case with a similar number of unknowns. This is due in part to the increase in the amount of preliminary computations (cf. Table III) and also the more complicated nature of the 3-D operators, especially the discretisation operator discussed in Section 2.2.

As for the two-dimensional cases, alternating line relaxation is able in each example to produce a convergent solution using less than the current upper limit of 10 iterations. However, the scheme is overall more demanding on time compared to the multi-grid algorithm incorporating the most efficient smoother available.

4. Concluding Remarks

A state-of-the-art multi-grid algorithm capable of solving elliptic partial differential equations in two and three dimensions has been developed and applied to a number of example problems based on situations occurring in oil reservoir simulation. These calculations have been selected to test the robustness of this particular choice of algorithm for handling problems exhibiting strong anisotropic properties and sharp jump discontinuities in the coefficient function $\mathbf{K}$ of the differential equation, Eq. (1.1). Moreover the overall aim of this paper has been to apply consistently a multi-grid algorithm employing specific relaxation and inter-

polation procedures to problems in two and three dimensions and to examine the relative success of such an algorithm in dealing with these problems.

Given a convergent relaxation procedure which has the appropriate smoothing property, multi-grid iteration is extremely fast at producing a solution to the majority of the representative examples considered in Section 3. The importance of first identifying the nature of the problem in question and then matching to it a suitable multi-grid sequence of operations is highlighted by those examples where the use of some relaxation procedures failed to readily provide a convergent solution.

Within the context of this study, the automatic prescription offers a robust and efficient multi-grid method for solving typical reservoir simulation problems. For anisotropic cases, the choice of relaxation scheme has a decisive bearing on efficiency and speed. The algorithm is particularly effective in two dimensions. In three dimensions, the gain in speed is partly off-set by the increase in computational time needed to set up the multi-level operators on each grid and by the degree of anisotropy and discontinuity possessed by the problems. Moreover, further work has shown that performance on two-dimensional problems is comparatively better [redacted] general trend is revealed whereby the algorithm used in this work is better suited to solving discontinuous and anisotropic problems in the plane rather than in space. It is believed, however, that further investigation and development of the three-dimensional interpolation and especially the relaxation procedure should remedy this situation.

ACKNOWLEDGMENTS

Discussions with Professor A. Brandt (Weizmann Institute of Science) and correspondence with Dr. J. E. Dendy, Jr. (Los Alamos National Laboratory) in the initial stages of this work are gratefully acknowledged. The work reported in this paper has been supported by the United Kingdom Department of Energy.

REFERENCES

1. R. E. ALCOUFFE, A. BRANDT, J. E. DENDY, JR., AND J. W. PAINTER, *SIAM J. Sci. Stat. Comput.* **2** (1981), 430–454.
2. K. AZIZ AND A. SETTARI, "Petroleum Reservoir Simulation," Applied Science, London, 1979.
3. A. BEHIE AND P. A. FORSYTH, *Appl. Math. Comput.* **13** (1983), 229–240.
4. A. BRANDT, *Math. Comput.* **31** (1977), 333–390.
5. A. BRANDT, "Multi-Level Adaptive Techniques (MLAT) for Partial Differential Equations: Ideas and Software" (J. R. Rice, Ed.), Mathematical Software III, pp. 277–318, Academic Press, New York, 1977.

6. A. BRANDT, "Guide to Multigrid Development" (W. Hackbusch and U. Trottenberg, Eds.), Proceedings Multigrid Methods Conference, Lecture Notes in Mathematics, Vol. 960, pp. 220–312, Springer-Verlag, Berlin, 1982.
7. J. R. DENDY, JR., *J. Comput. Phys.* **48** (1982), 366–386.
8. R. A. NICOLAIDES, *Math. Comput.* **31** (1977), 892–906.
9. H. S. PRICE AND K. H. COATS, *Soc. Pet. Eng. J.* June (1974), 295–308.