جلسه نهم - شبیه‌سازی مدرن مخازن نفتی

روش‌ها و ایده‌های

- ADI (دو بعدی)
- iADI (دو بعدی)
- ADI (سه بعدی، Douglas-Rachford)
- pSOR
- bSOR

6 Single-Phase Multidimensional Flow

برای تکمیل پایداری روش‌ها به کتب مرجع باید مراجعه نمود.

As yet a child,
Nor yet a fool to fame,
I lisped in numbers,
For the numbers came.

Alexander Pope

6.1 Introduction

In the previous chapter we treated a simple single-phase flow problem in 1-D. Also we considered two solution techniques, the explicit method and an implicit method (Crank-Nicolson) for solving the associated matrix problem. Both these techniques are examples of *direct* solution methods, i.e., no iteration was involved. Because of time-step limitations and the associated stability problem, completely explicit methods are rarely employed in reservoir simulation. More often we resort to implicit or semi-implicit methods, or possibly implicit methods combined with explicit computations. We can also employ iterative procedures as opposed to direct methods. However, the principal iterative techniques are deferred to this chapter where we discuss their application to multidimensional problems.

In a multidimensional environment, the basic approach is the same as that applied to the 1-D problem in chapter 5. That is, we begin with a partial differential equation (or collection of them) and replace it (them) with finite difference representations. This ultimately leads to a matrix problem, $Au = b$, to be solved. We must be concerned with the stability of the solution process, and convergence if the solution technique is iterative. In most cases, both are assured when the spectral radius of A or an associated matrix is bounded above by unity.

Before discussing specific single-phase flow problems, we first give a brief description of the grid systems most frequently employed in multidimensional problems. This is followed by a treatment of various alternating direction implicit procedures for formulating the algebraic problem. These require nothing more than a tridiagonal algorithm to achieve a solution. Attention is then turned toward the banded matrix problems which are typically encountered in reservoir simulation. Direct solution of these is discussed including a brief treatment of sparse matrix techniques. Finally, a discussion of two iterative methods, the strongly implicit procedure and successive overrelaxation is presented. Although the presentation is primarily in the context of a particular 2-D parabolic problem, these methods are equally applicable to any parabolic or elliptic problem in both 2-D and 3-D.

به تقسیم بندی معادلات PDE رجوع کنید.

6.2 Grid Systems and Boundary Conditions

As we have already indicated, one can relate the finite difference equations to a grid of rectangular blocks. Consider a reservoir with an arbitrary areal configuration as shown in Fig. 6.1. Now superimpose on this map the smallest possible rectangle that totally encompasses the reservoir. A grid consisting of orthogonal lines in each of the coordinate directions is then constructed forming a collection of rectangles bounded by the larger rectangle. Each smaller rectangle is called a *cell* or *block*. The blocks interior to the wiggly line in Fig. 6.2 are referred to as *active* blocks while those outside are *inactive*. Calculations are performed only for the active blocks in a reservoir simulator.

نکته: این تقسیم بندی مسائل و مقطعی پیاده سازی الگوریتمیک و کامپیوتری می باشد.

It is not necessary that the grid lines be equally spaced, i.e., block sizes need not be uniform. There are two basic grid types employed in reservoir work, block-centered grids and lattice-centered grids. For simplicity, we discuss these in a 2-D environment, but the same ideas apply to 3-D systems. In a block-centered grid, a point, P, is associated with the grid block center having indices (i,j), while the interfaces carry $(i \pm 1/2,$

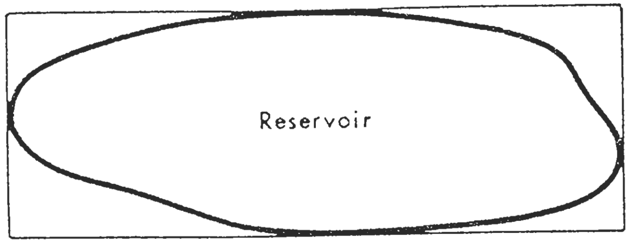


Fig. 6.1 Plan View of a Reservoir.

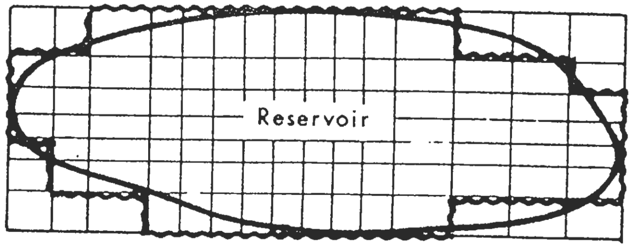


Fig. 6.2 Gridded Reservoir Map.

$j \pm 1/2)$ indices for a collection of blocks $i = 1, 2, \ldots, N_x$ in the x-direction, and $j = 1, 2, \ldots, N_y$ in the y-direction. In lattice-centered spacing, a point (i,j) is at the intersection of each grid line. Again, the grid is specified by the sequences, $i = 0, 1, 2, \ldots, N_x$ in the x-direction and $j = 0, 1, 2, \ldots, N_y$ in the y-direction. Block centers are at the half values for both indices. In Fig. 6.3 we illustrate block- and lattice-centered grids in 1-D.

The finite different representations of a system of partial differential equations is independent of the grid system employed; i.e., they are identical for both block- and lattice-centered grids. The only difference arises in the treatment of boundary conditions. The reason is the lattice-centered grid has boundaries coincident with the exterior calculation points, while the block-centered system has its boundaries one-half grid block beyond the exterior calculation points. Most commercial reservoir simulators employ block-centered grids, although in some applications, a lattice-type grid is preferable. For our purposes, it is sufficient to confine ourselves to block-centered grids. In this context we consider Dirichlet and Neumann-type boundary conditions. A Dirichlet boundary condition requires that the dependent variable be specified on the boundary. For example, a constant pressure (or temperature, etc.) boundary is a Dirichlet condition. On the other hand, if we specify the gradient of the dependent variable on the boundary, then we have a Neumann condition there. The most frequently

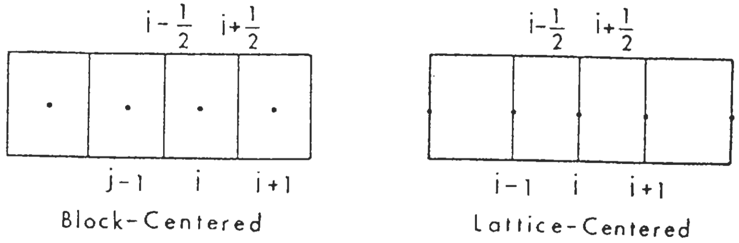


Fig. 6.3 Block- and Lattice-Centered Grid Systems.

employed Neumann condition in reservoir simulation is the "no-flow boundary;" i.e., if Φ is the flow potential, then $\nabla\Phi = 0$ on ∂R where ∂R is the closed boundary denoted by the wiggly line in Fig. 6.2. It is not unusual to have ***mixed boundary conditions,*** a Dirichlet condition over one part of ∂R and a Neumann condition over another. Reservoirs subject to water influx from an adjoining aquifer are examples. The water influx rate may be sufficient to maintain the potential at an essentially constant value in that part of the reservoir interfacing with the aquifer. This leads to a Dirichlet boundary condition there. Other parts of the reservoir may be sealed because of structural or stratigraphic trapping. Consequently, these represent no-flow boundaries of the Neumann type.

For block-centered grids, a Dirichlet condition is accounted for by simply specifying the value of the dependent variable (frequently pressure) at the block centers along that portion of the boundary affected. In other words, we neglect the fact that the block centers are not actually coincident with the reservoir boundaries. This approximation, of course, is improved if we use smaller grid spacings, i.e., a *refined grid*. However, this is not always practical in large reservoir problems. Furthermore, in global reservoir problems, efforts to do a better job on the boundaries are largely wasted since what goes on at interior points is little affected by boundary phenomena if the boundaries are sufficiently far removed. Indeed, in many instances, it makes no difference whether we specify a Neumann or Dirichlet condition: Where it does, we can always arbitrarily move the boundaries out to where their effects are not felt in the simulator, and concentrate on the areas of interest in the reservoir. The remarks above do not apply when we are examining single-well phenomena such as well-coning, etc., in a localized area. In such cases, realistic handling of boundary conditions can be very important and may entail a departure from block-centered schemes. Such considerations are treated by Settari and Aziz.[1]

A Neumann boundary condition for a block-centered grid system can be handled in one of two ways by: (1) using image blocks or (2) setting certain boundary coefficients to zero. Suppose for some 2-D region consisting of $N_x \times N_y$ blocks where $1 \le i \le N_x$ and $1 \le j \le N_y$, we require that $\nabla\Phi = 0$ on the far boundary in the x-direction, i.e., $\partial\Phi/\partial x = 0 \ \forall \ y$ there. Using a central difference to express this leads to

$$\Phi_{N_x-1,j} = \Phi_{N_x+1,j}, \quad 1 \le j \le N_y$$ (۱) روش ؟

where additional blocks with centers at $(N_x + 1, j)$ are introduced. Such blocks are *image blocks* occurring outside the reservoir, but are so-called because they reflect the value of the dependent variable at the interior points $(N_x - 1, j)$. More frequently, in reservoir simulation, we treat expressions of the form $\nabla \cdot [K\nabla\Phi]$ where K is a spatially- and possibly time-dependent coefficient. If we require $\nabla\Phi = 0$ on ∂R we can, in this case, accomplish

the same effect by setting $K = 0$ on ∂R. The nice thing about this approach is that one can specify flow or no-flow conditions by simply manipulating input data. (2) روش ؟

6.3 Slightly Compressible Flow

For a slightly compressible fluid in a homogeneous, isotropic, 2-D cartesian region where gravity is neglected, we have (see section 4.7)

$$\frac{\partial^2 p}{\partial X^2} + \frac{\partial^2 p}{\partial Y^2} = \frac{1}{\alpha}\frac{\partial p}{\partial t}; \quad \alpha \equiv \frac{k}{\phi\mu c} \tag{6.1}$$

which when normalized yields

$$u_{xx} + u_{yy} = u_t, \ t > 0, \ (x,y) \in R \tag{6.2}$$

where $R = [0, 1] \times [0, 1]$. On the far boundaries of R, i.e., at $x = 1$ $0 \le y \le 1$ and $y = 1$, $0 \le x \le 1$ we invoke a Dirichlet condition, $u = 0$, while on the near boundaries $x = 0$, $0 \le y \le 1$ and $y = 0$, $0 \le x \le 1$, Neumann conditions are specified, $u_x = 0$ and $u_y = 0$, respectively. We also require an initial condition, $u = u_0$ say, at $t = 0$ to get a well-posed problem. Using the difference operators defined in chapter 5 and subscripting the x- and y-parts, the explicit formulation of Eq. 6.2 is

$$(\Delta_x^2 u^n)/\Delta x^2 + (\Delta_y^2 u^n)/\Delta y^2 = (\Delta_t u)/\Delta t. \tag{6.3}$$

A stability analysis indicates that $\Delta t(\Delta x^{-2} + \Delta y^{-2})$ should be $\le$ ½ which is too restrictive for practical reservoir problems. If we appeal to the Crank-Nicolson scheme, then

$$\frac{\gamma_1}{2}[\Delta_x^2 u^{n+1} + \Delta_x^2 u^n] + \frac{\gamma_2}{2}[\Delta_y^2 u^{n+1} + \Delta_y^2 u^n] = \Delta_t u \tag{6.4}$$

مهم ترین روش ؟

where $\gamma_1 = \Delta t/\Delta x^2$ and $\gamma_2 = \Delta t/\Delta y^2$. This leads to a matrix problem of the type in Eq. 5.23; however, the coefficient matrix is *pentadiagonal* rather than tridiagonal. We defer treatment of such matrix problems until later. The matrix will be of order $N_x N_y$ which, for small computers, may be impractical to handle. An alternative approach is the fractional step method of Peaceman and Rachford[2] known as the Alternating Direction Implicit (ADI) procedure. The central idea is to consider a multidimensional problem as a collection of one-dimensional problems, each of which is solved over a

توضیح معادله 5.23 :

برای سیستم یک‌بعدی ($u_{xx}=u_t$)، اپراتورهای دیفرانس به شکل زیر تعریف می‌شوند:

$$\begin{cases} \Delta^2 u = u_{i-1} - 2u_i + u_{i+1} \\ \Delta_t u = u_i^{n+1} - u_i^n \end{cases}$$ (چون یک‌بعدی است، اندیس x نمی‌گذاریم)

فرمولاسیون صریح برای گسسته‌سازی PDE بالا : $\gamma \Delta^2 u^n = \Delta_t u$

فرمولاسیون Crank-Nich. برای گسسته‌سازی : $\gamma(\Delta^2 u^{n+1} + \Delta^2 u^n)/2 = \Delta_t u$

$\left(\gamma \triangleq \dfrac{\Delta t}{\Delta x^2} \right)$

با ارائه فرمولاسیون Crank-Nicholson، به منظور محاسبه معادله با دستگاه معادلاتی که مجهول‌های آن تنها مقادیر متغیر تابع در لحظه زمانی بعدی می‌باشند:

$$-\gamma u_{i-1}^{n+1} + 2(1+\gamma)u_i^{n+1} - \gamma u_{i+1}^{n+1} = \gamma u_{i-1}^n + 2(1-\gamma)u_i^n + \gamma u_{i+1}^n$$

چون مقدار متغیر تابع در لحظه بعدی را در هر گره نمی‌توان مستقیماً به دست آورد، این روش یک روش ضمنی است، لذا حل معادلات را باید با روش تکراری و یا حل دستگاهی به دست آورد:

$$A\underline{u} = \underline{d}$$

به طوریکه $\underline{u}$ بردار مجهولات است، یعنی مقادیر متغیرهای تابع در گره‌ها و در لحظه بعد.

$$A \triangleq \begin{bmatrix} b_1 & -c_1 & & & \bigcirc \\ -a_2 & b_2 & -c_2 & & \\ & -a_3 & b_3 & -c_3 & \\ & & \cdot & \cdot & \cdot \\ & \bigcirc & & \cdot \quad \cdot & \cdot \\ & & & -a_{N-1} & b_{N-1} \end{bmatrix}$$

$$\underline{d} \triangleq [\, d_1 \quad d_2 \quad d_3 \quad \dots \quad d_{N-1} \,]^T$$

$$b_i \triangleq 2(1+\gamma) \;,\quad a_i \triangleq \gamma \;,\quad c_i \triangleq \gamma \;,\quad i = 1, 2, \dots, N-1$$

دقت کنید ماتریس A، در حالت یک‌بعدی، یک ماتریس سه قطری است.

fraction of a time step.† The associated matrix problems are always tridiagonal. This is discussed next.

ایده ADI را در روشهای نظیر FEM نیز بکار برده‌اند.

6.4 Alternating Direction Implicit Procedure

Instead of Eq. 6.3, we express one of the coordinate directions implicitly leaving the other explicit and consider that time is advanced over half a time step. Then the roles of the implicit and explicit parts are interchanged to complete the time step. We refer to the spatial derivative evaluated implicitly as the *sweeping direction.* Thus, if the x-part is implicit,

$$(\Delta_x^2 u^{n+1/2})/\Delta x^2 + (\Delta_y^2 u^n)/\Delta y^2 = \frac{2}{\Delta t}(u^{n+1/2} - u^n) \tag{6.5}$$

for the x-sweep, and

$$\frac{\partial u}{\partial t} \simeq \frac{u^{n+1/2} - u^{n}}{(\Delta t/2)} = \frac{u^{n+1} - u^{n+1/2}}{(\Delta t/2)}$$

$$(\Delta_x^2 u^{n+1/2})/\Delta x^2 + (\Delta_y^2 u^{n+1})/\Delta y^2 = \frac{2}{\Delta t}(u^{n+1} - u^{n+1/2}) \tag{6.6}$$

for the y-sweep. Eqs. 6.5 and 6.6 lead to matrix problems of the form

$$\mathbf{H}\mathbf{u}^{n+1/2} = \mathbf{d}_x \tag{6.7}$$

$$\mathbf{V}\mathbf{u}^{n+1} = \mathbf{d}_y \tag{6.8}$$

where both $\mathbf{H}$ and $\mathbf{V}$ are tridiagonal matrices. In $\mathbf{H}$ the diagonal term is $2(1+\gamma_1)$ while the off-diagonals are $-\gamma_1$. In $\mathbf{V}$ the diagonal and off-diagonals are $2(1+\gamma_2)$ and $-\gamma_2$, respectively. Also $\mathbf{d}_x$ contains all terms in Eq. 6.5 evaluated at time level n while $\mathbf{d}_y$ contains those in Eq. 6.6 at time-level $n + ½$. The solution process is started using the initial condition, $u = u_0$ at time level zero, i.e., $n = 0$, to determine the right-hand side of Eq. 6.7. Solution of Eq. 6.7 for $\mathbf{u}^{n+1/2}$ permits us to evaluate $\mathbf{d}_y$ for use in the y-sweep. Consequently, alternately solving Eqs. 6.7 and 6.8 repeatedly advances the solution in time.

A more realistic application of ADI is to single-phase flow in anisotropic, heterogeneous media containing source (or sink) terms. Consider the problem

$$\frac{\partial}{\partial x}\left(\frac{k_x}{\mu}\frac{\partial p}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{k_y}{\mu}\frac{\partial p}{\partial y}\right) + f(x,y,t) = \phi c\frac{\partial p}{\partial t} \tag{6.9}$$

† This concept has been exploited in contexts other than finite difference methods.[3-6]

where $(x,y)\epsilon R$, $t > 0$. On R, we define partitions

$$\Pi_x\colon 0 < x_1 < x_2 < x_3 < \ldots < x_{N_x}$$

$$\Pi_y\colon 0 < y_1 < y_2 < y_3 < \ldots < y_{N_y}$$

such that a collection of rectangular blocks is defined, not necessarily equally spaced. If the grid is block-centered, then

$$x_i = ½(x_{i-1/2} + x_{i+1/2}),\ \Delta x_i = x_{i+1/2} - x_{i-1/2},\ i = 1, 2, \ldots, N_x$$
$$y_j = ½(y_{j-1/2} + y_{j+1/2}),\ \Delta y_j = y_{j+1/2} - y_{j-1/2},\ j = 1, 2, \ldots, N_y$$

A second order correct approximation for the derivatives in Eq. 6.9 is

$$\frac{\partial}{\partial s}\left(a_s\frac{\partial p}{\partial s}\right) = \frac{(2a_s)_{k+1/2}(p_{k+1} - p_k)}{\Delta s_k(\Delta s_{k+1} + \Delta s_k)} - \frac{(2a_s)_{k-1/2}(p_k - p_{k-1})}{\Delta s_k(\Delta s_k + \Delta s_{k-1})}$$

where $s = x$ or y, $k = i$ or j and $a_s = k_x/\mu$ or k_y/μ.

Define $(A_s)_{k\pm 1/2} \equiv (2a_s)_{k\pm1/2}/(\Delta s_k + \Delta s_{k\pm1})$ then,

$$\frac{\partial}{\partial s}\left(a_s\frac{\partial p}{\partial s}\right) = \{(A_s)_{k+1/2}(p_{k+1} - p_k) - (A_s)_{k-1/2}(p_k - p_{k-1})\}/\Delta s_k$$
$$\equiv \Delta_s A_s \Delta_s p/\Delta s_k. \tag{6.10}$$

With the difference notation in Eq. 6.10, the finite difference equation for Eq. 6.9 is

$$(\Delta_x A_x \Delta_x p)/\Delta x_i + (\Delta_y A_y \Delta_y p)/\Delta y_j + f(x,y,t) = \frac{\phi c}{\Delta t}(p_{ij}^{n+1} - p_{ij}^{n}). \tag{6.11}$$

If we multiply Eq. 6.11 by $\Delta x_i \Delta y_j$; then,

$$\Delta_x T_x \Delta_x p + \Delta_y T_y \Delta_y p + q_{ij} = \frac{\alpha}{\Delta t}(p_{ij}^{n+1} - p_{ij}^{n}) \tag{6.12}$$

where $\alpha \equiv c\Delta x_i \Delta y_j \phi$; $q_{ij} \equiv \Delta x_i \Delta y_j f(x,y,t)$,

$$\Delta_x T_x \Delta_x p \equiv (T_x)_{i+1/2}(p_{i+1,j} - p_{i,j}) - (T_x)_{i-1/2}(p_{i,j} - p_{i-1,j}),$$

$(T_x)_{i+1/2} \equiv (A_x)_{i+1/2}\Delta y_j$, and $(T_y)_{j-1/2} \equiv (A_y)_{j-1/2}\Delta x_i$, i.e.,

$$(T_x)_{i+1/2} \equiv \frac{2(k_x/\mu)_{i+1/2}\Delta y_j}{\Delta x_i + \Delta x_{i+1}},$$ for example.

The T's in the above expressions are called *transmissibilities* or *transmissivities*. (The latter term is most frequently used by hydrologists, while the former is commonly used in the oil industry, perhaps incorrectly.[7]) One should observe the analogous notation in Eqs. 6.9 and 6.12; i.e., the difference operator for $\partial/\partial x$ becomes Δ_x and similarly $\partial/\partial y$ has the analog Δ_y. Thus, if $k_x/\mu = k_y/\mu$ = a constant, we have

$$\Delta_x^2 p + \Delta_y^2 p + q_{ij}^* = \frac{\beta}{\Delta t}(p_{ij}^{n+1} - p_{ij}^n) \tag{6.13}$$

for the linear problem where $\beta \equiv (\phi\mu c/k)(\Delta x_i \Delta y_j)$, and $q_{ij}^* \equiv q_{ij}\mu/k$. Frequently we use the operator $\Delta T \Delta p$ to denote the spatial difference operators in 1-, 2-, or 3-D. Thus it can represent the sum of the two left-hand side terms in Eq. 6.12. In 3-D it includes an additional term similar to those above for the z-direction. Thus a compact form for Eq. 6.12 is

$$\Delta T \Delta p + q_{ij} = \frac{\alpha}{\Delta t}\Delta_t p$$

where $\Delta_t p$ denotes the time difference in Eq. 6.12.

The ADI formulation of Eq. 6.9 is

[illegible]

$$\boxed{\begin{aligned} &x\text{-sweep: } \Delta_x T_x \Delta_x p^{n+1/2} + \Delta_y T_y \Delta_y p^n + q^n = \frac{2\alpha}{\Delta t}(p^{n+1/2} - p^n) && (6.14)\\ &y\text{-sweep: } \Delta_x T_x \Delta_x p^{n+1/2} + \Delta_y T_y \Delta_y p^{n+1} + q^n = \frac{2\alpha}{\Delta t}(p^{n+1} - p^{n+1/2}) && (6.15)\end{aligned}}$$

where indices i and j on the q-terms and the right-hand sides are suppressed. Eq. 6.14 can be written as

x-sweep
[illegible]

$$\boxed{-a_i p_{i-1}^{n+1/2} + b_i p_i^{n+1/2} - c_i p_{i+1}^{n+1/2} = d_i} \tag{6.16}$$

where $a_i = (T_x)_{i-1/2}$, $b_i = \left[(T_x)_{i+1/2} + (T_x)_{i-1/2} + \frac{2\alpha}{\Delta t}\right]$, $c_i = (T_x)_{i+1/2}$,

and $d_i = (T_y)_{j-1/2}p_{j-1}^n - \left[(T_y)_{j+1/2} + (T_y)_{j+1/2} - \frac{2\alpha}{\Delta t}\right]p_{ij}^n + (T_y)_{j+1/2}p_{j+1}^n + q_{ij}^n$.

Eq. 6.15 becomes

y-sweep

$$\boxed{-\hat{a}_j p_{j-1}^{n+1} + \hat{b}_j p_{ij}^{n+1} - \hat{c}_j p_{j+1}^{n+1} = \hat{d}_j} \tag{6.17}$$

where $\hat{a}_j = (T_y)_{j-1/2}$, $\hat{b}_j = \left[(T_y)_{j-1/2} + (T_y)_{j+1/2} + \frac{2\alpha}{\Delta t}\right]$, $\hat{c}_j = (T_y)_{j+1/2}$,

and $\hat{d}_j = (T_x)_{i-1/2}p_{i-1}^{n+1/2} - \left[(T_x)_{i-1/2} + (T_x)_{i+1/2} - \frac{2\alpha}{\Delta t}\right]p_{ij}^{n+1/2} + (T_x)_{i+1/2}p_{i+1}^{n+1/2} + q_{ij}^{n+1/2}$.

In the expressions above, with the exception of terms referring to the point (i,j), only the "fastest moving" index is shown to avoid a clutter of subscripts. By fastest moving index, we mean the index that runs through all its values for *fixed* values of the others. For example, for the x-sweep, i is the fastest moving index since for a fixed j, $i = 1, 2, \ldots, N_x$. For simplicity of notation we follow this practice throughout. Note that both Eqs. 6.16 and 6.17 lead to tridiagonal matrix problems for each sweep as anticipated. Unlike the matrix problems associated with Eqs. 6.5 and 6.6, however, the matrix coefficients in this case are not constants but are space-dependent functions. Later we shall see that the treatment of multiphase flow problems involves space and time-dependent matrix coefficients.

The ADI procedure of Peaceman and Rachford is unconditionally stable as will be shown. However, truncation errors can become prohibitive for large Δt which severely restricts the time-step size in many practical applications. Furthermore, when the transmissibilities are strongly contrasting in each of the coordinate directions, convergence is difficult if not impossible to achieve. Again, in treating some problems involving moving fronts with ADI, a substantial "ringing effect" is observed in the vicinity of the front. This produces oscillations in the dependent variable about some fixed value. These oscillations are not instabilities in the true sense of the word, since they usually dampen with time. Actually, they are associated with "Crank-Nicolson noise" insofar as Peaceman-Rachford ADI is a perturbed form of the Crank-Nicolson procedure (see section 6.6). There are techniques for controlling the oscillations[8] and even removing them entirely by smoothing procedures.[9] Nevertheless, the other disadvantages of ADI are such that it is now rarely, if ever, used in commercial simulators. Some, however, provide an option to use an iterative ADI approach (IADI) described next.

6.5 Iterative Alternating Direction Implicit Method

The procedure we describe is that of Peaceman and Rachford.[2] Later we discuss variations of this technique. Instead of treating each sweep as an advancement in time level, one can express the pertinent equations as advancements in *iteration levels* to get from n to $n + 1$. Thus, Eqs. 6.14 and 6.15 can be written as

$$\Delta_x T_x \Delta_x p^{k+1/2} + \Delta_y T_y \Delta_y p^k - \frac{2\alpha}{\Delta t} p^{k+1/2} = -\frac{2\alpha}{\Delta t} p^n - q^n \qquad (6.18)$$

$$\Delta_x T_x \Delta_x p^{k+1/2} + \Delta_y T_y \Delta_y p^{k+1} - \frac{2\alpha}{\Delta t} p^{k+1} = -\frac{2\alpha}{\Delta t} p^n - q^n \qquad (6.19)$$

Note in Eq. 6.19 $- 2\alpha/\Delta t\ p^n$ appears rather than $- 2\alpha/\Delta t\ p^{n+1/2}$ as in Eq. 6.15. To compensate for this slight modification, an appropriate multiple of the difference in iterates is added to each side of Eqs. 6.18 and 6.19 to arrive at the iterative finite difference scheme:

$$\Delta_x T_x \Delta_x p^{k+1/2} + \Delta_y T_y \Delta_y p^k - \frac{2\alpha}{\Delta t} p^{k+1/2} = H_k(p^{k+1/2} - p^k) - C \qquad (6.20)$$

$$\Delta_x T_x \Delta_x p^{k+1/2} + \Delta_y T_y \Delta_y p^{k+1} - \frac{2\alpha}{\Delta t} p^{k+1} = H_k(p^{k+1} - p^{k+1/2}) - C \qquad (6.21)$$

where H_k is a product of an iteration parameter† and a normalizing factor and $C = 2\alpha/\Delta t\ p^n + q^n$. If in Eqs. 6.20 and 6.21, the implicit terms are placed on the left-hand sides, and the explicit terms on the right-hand sides, then again the problem takes on a tridiagonal form.

The normalizing factor used to formulate H_k is the sum of the transmissibilities around the four faces of an (i,j)-block. Thus we write $H_k = \sigma_k\,(\Sigma T)$ where $\Sigma T \equiv (T_x)_{i+1/2,j} + (T_x)_{i-1/2,j} + (T_y)_{i,j+1/2} + (T_y)_{i,j-1/2}$ and σ_k is an iteration parameter. The subscript, k, is assigned to σ since normally it will change from iteration to iteration. In practice, a set of parameters are employed, $\sigma_1, \sigma_2, \ldots, \sigma_K$, where $\sigma_1 < \sigma_2 < \ldots < \sigma_K$ and one cycles through them from the smallest to the largest (or sometimes vice versa) and repeats the cycle, in whole or in part, until convergence is achieved. The parameters should be geometrically spaced, i.e., $\sigma_{k+1}/\sigma_k = \alpha$, a constant. If a total of K parameters are chosen per cycle, then $\sigma_K/\sigma_1 = \alpha^{K-1}$. If σ_1, σ_K (the minimum and maximum parameters) and the number of parameters per cycle are known, then α may be calculated from

$$\ln(\alpha) = \frac{\ln(\sigma_K/\sigma_1)}{K-1}. \qquad (6.22)$$

† Iteration parameters are real numbers used to accelerate convergence rates.

Usually K is 4 or 5 for a small range on the σ_k (e.g., .01 to 2) and 6 to 8 for a large range (e.g., .0001 to 2).

Unfortunately, there is no theoretical basis for selecting iteration parameters for many practical simulation problems, except for highly idealized cases. For example, if N_x and N_y are sufficiently large and the transmissibilities T_x and T_y are uniform, but not necessarily equal, then the minimum parameter is given by

$$\sigma_1 = \text{Min}\left\{\frac{\pi^2}{2N_x^2}\left(\frac{1}{1 + T_y/T_x}\right), \frac{\pi^2}{2N_y^2}\left(\frac{1}{1 + T_x/T_y}\right)\right\}. \qquad (6.23)$$

If $T_x = T_y$, the maximum is 1 and if $T_x >> T_y$ or $T_y >> T_x$, the maximum is 2. For equally spaced grid points, $T_x = k_x\ \Delta y/\Delta x$, $T_y = k_y\ \Delta x/\Delta y$, then

$$\sigma_1 = \min\left\{\frac{\pi^2}{2N_x^2}\frac{1}{1 + \dfrac{k_y\Delta x^2}{k_x\Delta y^2}}, \frac{\pi^2}{2N_y^2}\frac{1}{1 + \dfrac{k_x\Delta y^2}{k_y\Delta x^2}}\right\}. \qquad (6.24)$$

For an areal (2-D) problem, generally $\Delta x \approx \Delta y$ and if $k_x = k_y$, then the two terms in Eq. 6.24 are equal. For cross-sectional problems, however, generally $N_x >> N_y$, $\Delta x >> \Delta y$ and the first term is smaller.

In cases where the transmissibilities are highly variable, Eqs. 6.23 and 6.24 can serve only as rough guides. One could conceivably compute σ_1 for each (i,j)-block using Eq. 6.23 and then select the minimum from that set. The maximum value will always be between 2 and a number slightly less than 1.[10] The convergence rate is usually insensitive to its actual magnitude within this range. However, there is extreme sensitivity to σ_1 for some problems. A change from 0.005 to 0.0001 may be critical and cause divergence. When this occurs 3 or 4 trial runs may be necessary to determine a σ_k set that will be satisfactory.

Frequently as convergence is approached, the difference between successive iterates becomes so small that significant digits are cancelled on a computer. This occurs sooner on machines with short word length. This error can be minimized by solving for the displacement in each iteration rather than for the iterate itself. Thus if we define $PX \equiv p^{k+1/2} - p^k$ and $PY \equiv p^{k+1} - p^k$ then, Eqs. 6.20 and 6.21 become

$$\Delta_x T_x \Delta_x PX - \left(\frac{2\alpha}{\Delta t} + H_k\right) PX = \frac{2\alpha}{\Delta t} p^k - \Delta T \Delta p^k - C \qquad (6.25)$$

and, $$\Delta_y T_y \Delta_y PY - \left(\frac{2\alpha}{\Delta t} + H_k\right) PY = \frac{2\alpha}{\Delta t} p^k - \Delta T \Delta p^k - C - \Delta_x T_x \Delta_x PX - H_k PX. \qquad (6.26)$$

Let $G \equiv 2\alpha/\Delta t\ p^k - \Delta T \Delta p^k - C$ and notice that the right-hand side of Eq. 6.26 can be expressed in terms of PX and G. Since $G = \Delta_x T_x \Delta_x PX - (2\alpha/\Delta t + H_k)PX$ from Eq. 6.25, we get the residual form,

$$x\text{-sweep: } \Delta_x T_x \Delta_x PX - \left(\frac{2\alpha}{\Delta t} + H_k\right) PX = G \tag{6.27}$$

$$y\text{-sweep: } \Delta_y T_y \Delta_y PY - \left(\frac{2\alpha}{\Delta t} + H_k\right) PY = -2\left(\frac{\alpha}{\Delta t} + H_k\right) PX \tag{6.28}$$

Now consider the implicit finite difference formulation of the original problem:

$$\Delta T \Delta p^{n+1} + q^n = \frac{2\alpha}{\Delta t}(p^{n+1} - p^n). \tag{6.29}$$

Since $C = q^n + 2\alpha/\Delta t\ p^n$, then we have

$$-\frac{2\alpha}{\Delta t} p^{n+1} + \Delta T \Delta p^{n+1} + C = 0. \tag{6.30}$$

Notice G is identical to Eq. 6.30 using the k^{th} iterate. If convergence to the exact solution at time level $n+1$ is achieved then G would be equal to zero. This observation can be used as a closure criterion. Let the residual, $R_{i,j}$ be given by

$$R_{ij} = \Delta T \Delta p^k - \frac{2\alpha}{\Delta t} p^k + C, \tag{6.31}$$

and invoke the following closure criteria:

(1) $\epsilon_1 = \sum\limits_{i,j} |R_{ij}|$
$= \sum\limits_{i=1}^{N_x} \sum\limits_{j=1}^{N_y} |R_{ij}|$ (absolute residual)

(2) $\epsilon_2 = \dfrac{\sum\limits_{i,j} |R_{ij}|}{\sum\limits_{i,j} |Q_{ij}|}$ (normalized residual)

Usually $\epsilon_1 \le 0.001$ and $\epsilon_2 \le .05$ are satisfactory for closure.

6.6 Stability and Accuracy of ADI Methods

We concern ourselves here only with the stability and truncation error of the noniterative ADI methods of Peaceman-Rachford. The iterative procedure will generally display the same stability and accuracy characteristics as the direct method.

Because of the spatial variation of permeability in Eq. 6.9, a stability analysis is difficult to achieve without some simplification. For this reason, attention is directed to a von Neumann stability analysis of Eq. 6.2 to illustrate the ideas involved. Let the error component be given by

$$\epsilon_{ij}^n = \zeta^n e^{Jpi\Delta x} e^{Jqj\Delta y} \tag{6.32}$$

analogous to Eq. 5.34 where $p = k\pi/N_x\Delta x$ and $q = k\pi/N_y\Delta y$, $k = 1, 2, 3, \ldots$, and we require that $|\zeta| \le 1$ for stability. Since the error component in Eq. 6.32 is propagated by the difference equations (Eqs. 6.5 and 6.6), substitution of ϵ_{ij} into the latter two equations leads to

$$\zeta = \frac{1 - 4\gamma \sin^2(q\Delta y/2)}{1 + 4\gamma \sin^2(p\Delta x/2)} \tag{6.33}$$

for the x-sweep, and

$$\zeta = \frac{1 - 4\gamma \sin^2(p\Delta x/2)}{1 + 4\gamma \sin^2(q\Delta y/2)} \tag{6.34}$$

for the y-sweep. In arriving at Eqs. 6.33 and 6.34 each sweep is considered as a full step in time; i.e., we omitted the 2's in Eqs. 6.5 and 6.6, and assumed the x-sweep takes us from level n to $n+1$, and the y-sweep from $n+1$ to $n+2$. Also, $\gamma = \Delta t/\Delta x^2 = \Delta t/\Delta y^2$. When $\gamma > ½$, $p = 1$, $q = N_y - 1$ then $|\zeta| > 1$, i.e., Eq. 6.5, becomes unstable. Similarly, Eq. 6.6 is unstable for certain values of the same parameters. We demonstrate that $|\zeta| > 1$ for Eq. 6.33 only.

Consider some function $f(x,\hat{x}) = (1 - x)/(1 + \hat{x})$ where $\hat{x}$ is fixed. Then if $0 < x_1 \le x \le x_2$, it follows that $f_x < 0\ \forall\ x\ \epsilon[x_1,x_2]$ and this implies that $f_{\max}$ occurs at one of the end-points, i.e.,

$$f_{\max} = \max\left\{\left|\frac{1-x_1}{1+\hat{x}}\right|, \left|\frac{1-x_2}{1+\hat{x}}\right|\right\}.$$

In Eq. 6.33, set $\gamma = 1$ and suppose $\Delta x = \Delta y = \pi/N$ where $N_x = N_y \equiv N$; thus,

$$\zeta = \frac{1 - 4\sin^2\left(\frac{k\pi}{2N}\right)}{1 + 4\sin^2\left(\frac{\pi}{2N}\right)}, \; k = 1, 2, \ldots, N-1 \qquad (6.35)$$

where $N > 1$. Now $x_1 = \hat{x} = 4\sin^2(\pi/2N)$ and

$$x_2 = 4\sin^2\left[\frac{\pi(N-1)}{2N}\right] = 4\cos^2\left(\frac{\pi}{2N}\right).$$

Consequently,

$$|\zeta|_{\max} = \max\left(\left|\frac{1 - 4\sin^2(\pi/2N)}{1 + 4\sin^2(\pi/2N)}\right|, \left|\frac{1 - 4\cos^2(\pi/2N)}{1 + 4\sin^2(\pi/2N)}\right|\right). \qquad (6.36)$$

Moreover, $\cos^2(\pi/2N) \geq \sin^2(\pi/2N) \; \forall \; N > 1$; thus, $|\zeta|_{\max}$ is the second term in the curly brackets in Eq. 6.36 and this is greater than one for N sufficiently large.

If we consider Eqs. 6.33 and 6.34 as the results of half-steps in time, then the right-hand side of Eq. 6.33 $= \zeta^{n+1/2}/\zeta^n$ and the right-hand side of Eq. 6.34 $= \zeta^{n+1}/\zeta^{n+1/2}$ such that

$$\zeta = \frac{1 - 2\gamma\sin^2(q\Delta y/2)}{1 + 2\gamma\sin^2(p\Delta x/2)} \; \frac{1 - 2\gamma\sin^2(p\Delta x/2)}{1 + 2\gamma\sin^2(q\Delta y/2)}. \qquad (6.37)$$

One can readily see in this case that $|\zeta| \leq 1$ for $\gamma > 0$. Clearly, the individual sweeps of ADI can become unstable, while the composite effect of the x and y-sweeps yields an unconditionally stable procedure.

As an alternative to a truncation error analysis, we relate the ADI procedure to the Crank-Nicolson scheme via an *overall equation* as done by Douglas[11], and thereby determine the truncation error much more simply. For example, subtract Eq. 6.14 from Eq. 6.15 and get

$$\Delta_y T_y \Delta_y (p^{n+1} - p^n) = \frac{2\alpha}{\Delta t}(p^{n+1} + p^n - 2\,p^{n+1/2})$$

from which

$$p^{n+1/2} = \tfrac{1}{2}(p^{n+1} + p^n) - \frac{\Delta t}{4\alpha}\Delta_y T_y \Delta_y (p^{n+1} - p^n). \qquad (6.38)$$

Adding Eqs. 6.14 and 6.15 yields

$$2\Delta_x T_x \Delta_x p^{n+1/2} + \Delta_y T_y \Delta_y (p^{n+1} + p^n) + 2\,q^n = \frac{2\alpha}{\Delta t}(p^{n+1} - p^n). \qquad (6.39)$$

Now, substitute Eq. 6.38 in Eq. 6.39 and get the overall equation,

$$\Delta_x T_x \Delta_x \frac{(p^{n+1} + p^n)}{2} + \Delta_y T_y \Delta_y \frac{(p^{n+1} + p^n)}{2} + q^n$$
$$= \frac{\alpha}{\Delta t}(p^{n+1} - p^n) + \frac{\Delta t}{4\alpha}\Delta_x T_x \Delta_x \Delta_y \, T_y \Delta_y \,(p^{n+1} - p^n) \qquad (6.40)$$

which is equivalent to the Crank-Nicolson approximation augmented by a perturbation term (the last term on the right-hand side). Consequently the error is $0(\Delta x^2 + \Delta y^2 + \Delta t^2)$ and convergence is guaranteed when $\Delta s \rightarrow 0$, $s = x$, y and t.

6.7 Flow Problems in 3-D

An attempt to extend the Peaceman-Rachford technique to single-phase flow in 3-D, shows that the composite of three sweeps can lead to instabilities. For example, for the problem

$$u_{xx} + u_{yy} + u_{zz} = u_t \qquad (6.41)$$

the *amplification factor*, ζ, as determined by a von Neumann analysis is

$$\zeta = \frac{(a + b + c)(b - 3)}{(1 + a)(1 + b)(1 + c)} + 1 \qquad (6.42)$$

where

$$a = 4\gamma_1 \sin^2(p\Delta x/2); \; \gamma_1 = \Delta t/\Delta x^2$$

$$b = 4\gamma_2 \sin^2(q\Delta y/2); \; \gamma_2 = \Delta t/\Delta y^2$$

$$c = 4\gamma_3 \sin^2(r\Delta z/2); \; \gamma_3 = \Delta t/\Delta z^2$$

When $b > 3$, the system becomes unstable.

Douglas and Rachford[12] developed an unconditionally stable ADI scheme for 3-D which we illustrate for Eq. 6.41. The formulation is

$$x: \quad \Delta_x^2 u^{n+1/3} + \Delta_y^2 u^n + \Delta_z^2 u^n = \frac{u^{n+1/3} - u^n}{\Delta t} \qquad (6.43)$$

$$y: \quad \Delta_y^2 u^{n+2/3} = \Delta_y^2 u^n + (u^{n+2/3} - u^{n+1/3})/\Delta t \qquad (6.44)$$

$$z: \quad \Delta_z^2 u^{n+1} = \Delta_z^2 u^n + (u^{n+1} - u^{n+2/3})/\Delta t \qquad (6.45)$$

for each of the three sweeps. If we solve Eq. 6.44 for $\Delta_y^2 u^n$ and substitute into Eq. 6.43, we get an alternative expression for the y-sweep. Similarly, if we use Eq. 6.45 to replace $\Delta_z^2 u^n$ in the new y-sweep equation, we get a new expression for the z-sweep. Thus, the alternate forms are

$$x: \quad \Delta_x^2 u^{n+1/3} + \Delta_y^2 u^n + \Delta_z^2 u^n = (u^{n+1/3} - u^n)/\Delta t \tag{6.46}$$

$$y: \quad \Delta_x^2 u^{n+1/3} + \Delta_y^2 u^{n+2/3} + \Delta_z^2 u^n = (u^{n+2/3} - u^n)/\Delta t \tag{6.47}$$

$$z: \quad \Delta_x^2 u^{n+1/3} + \Delta_y^2 u^{n+2/3} + \Delta_z^2 u^{n+1} = (u^{n+1} - u^n)/\Delta t \tag{6.48}$$

Eqs. 6.46–6.48 differ from the Peaceman-Rachford formulation only in the right-hand sides. In the equations above, the time displacements are measured from u^n in all three sweeps rather than from $u^{n+1/3}$ and $u^{n+2/3}$ in the y- and z-sweeps, respectively, of the Peaceman-Rachford method. Also, division is by the total time step, Δt.

This technique can be applied to 2-D problems, however, the time truncation error is greater. The overall equation is, in this case

$$(\Delta_?^2 + \Delta_y^2) u^{n+1} = (u^{n+1} - u^n)/\Delta t + \frac{\Delta t}{2} \Delta_x^2 \Delta_y^2 (u^{n+1} - u^n). \tag{6.49}$$

Consequently, the truncation error is $0(\Delta x^2 + \Delta y^2 + \Delta t)$. Clearly, the Peaceman-Rachford method is superior for 2-D problems.

An iterative Douglas-Rachford scheme for the 3-D problem analogous to Eq. 6.9 has the form

$$x: \ \Delta_x T_x \Delta_x p^{k+1/3} + \Delta_y T_y \Delta_y p^k + \Delta_z T_z \Delta_z p^k - \frac{\alpha}{\Delta t} p^{k+1/3} = H_k(p^{k+1/3} - p^k) - C \tag{6.50}$$

$$y: \ \Delta_x T_x \Delta_x p^{k+1/3} + \Delta_y T_y \Delta_y p^{k+2/3} + \Delta_z T_z \Delta_z p^k - \frac{\alpha}{\Delta t} p^{k+2/3} = H_k(p^{k+2/3} - p^k) - C \tag{6.51}$$

$$z: \ \Delta_x T_x \Delta_x p^{k+1/3} + \Delta_y T_y \Delta_y p^{k+2/3} + \Delta_z T_z \Delta_z p^{k+1} - \frac{\alpha}{\Delta t} p^{k+1} = H_k(p^{k+1} - p^k) - C \tag{6.52}$$

where $\alpha \equiv \phi c \Delta x_i \Delta y_j \Delta z_k$ and $C \equiv (\alpha/\Delta t) p^n + q^n$. This, like the direct Douglas-Rachford method, is unconditionally stable with truncation error of $0(\Delta x^2 + \Delta y^2 + \Delta z^2 + \Delta t)$, i.e., it is second order correct in space and first order correct in time.

To establish a relationship between Peaceman-Rachford ADI and Douglas-Rachford ADI, consider half a time step in the Douglas-Rachford method and extrapolate to the full time step. For example, the 2-D Douglas-Rachford formulation for Eq. 6.2 is

$$\Delta_x^2 u^{n+1/2} + \Delta_y^2 u^n = \frac{u^{n+1/2} - u^n}{\Delta t} \tag{6.53}$$

$$\Delta_y^2 u^{n+1} = \Delta_y^2 u^n + \frac{u^{n+1} - u^{n+1/2}}{\Delta t} \tag{6.54}$$

If we make the replacements, $u^{n+1/4} \rightarrow u^{n+1/2}$, $u^{n+1/2} \rightarrow u^{n+1}$ and $\Delta t/2 \rightarrow \Delta t$ then,

$$\Delta_x^2 u^{n+1/4} + \Delta_y^2 u^n = \frac{(u^{n+1/4} - u^n)}{\Delta t/2} \tag{6.55}$$

$$\Delta_y^2 u^{n+1/2} = \Delta_y^2 u^n + \frac{u^{n+1/2} - u^{n+1/4}}{\Delta t/2} \tag{6.56}$$

The extrapolation to the full time step is done using

$$u^{n+1} = u^n + 2(u^{n+1/2} - u^n) \tag{6.57}$$

from which we get

$$u^{n+1/2} = (u^{n+1} + u^n)/2 \tag{6.58}$$

$$u^{n+1/2} - u^n = (u^{n+1} - u^n)/2 \tag{6.59}$$

Now substitute Eqs. 6.58 and 6.59 into Eq. 6.49 to get

$$(\Delta_x^2 + \Delta_y^2)(u^{n+1} + u^n)/2 = (u^{n+1} - u^n)/\Delta t + \frac{\Delta t}{4} \Delta_x^2 \Delta_y^2 (u^{n+1} - u^n) \tag{6.60}$$

which is the overall Peaceman-Rachford equation analogous to Eq. 6.40.

Brian[13] used this idea for a 3-D formulation that was both second order correct in space and time. Namely, he formulated the Douglas-Rachford method over half a time step and extrapolated to the $(n+1)^{st}$ time level. Consequently, for Eq. 6.41.

$$x: \quad \Delta_x^2 u^{n+1/6} + \Delta_y^2 u^n + \Delta_z^2 u^n = 2(u^{n+1/6} - u^n)/\Delta t \tag{6.61}$$

$$y: \quad \Delta_y^2 u^{n+1/3} = \Delta_y^2 u^n + 2(u^{n+1/3} - u^{n+1/6})/\Delta t \qquad (6.62)$$

$$z: \quad \Delta_y^2 u^{n+1/2} = \Delta_y^2 u^n + 2(u^{n+1/2} - u^{n+1/3})/\Delta t \qquad (6.63)$$

Again using Eqs. 6.57–6.59, we get

$$\Delta_x^2 + \Delta_y^2 + \Delta_z^2(u^{n+1} + u^n)/2 = (u^{n+1} - u^n)/\Delta t + \text{perturbation term} \qquad (6.64)$$

which is a variation of Crank-Nicolson's formulation with truncation error of $0(\Delta x^2 + \Delta y^2 + \Delta z^2 + \Delta t^2)$. Douglas[14] suggested formulating the Crank-Nicolson procedure directly for the 3-D problem; i.e.,

$$x: \quad \Delta_x^2(u^{n+1/3} + u^n)/2 + \Delta_y^2 u^n + \Delta_z^2 u^n = (u^{n+1/3} - u^n)/\Delta t \qquad (6.65)$$

$$y: \quad \Delta_x^2(u^{n+1/3} + u^n)/2 + \Delta_y^2(u^{n+2/3} + u^n)/2 + \Delta_z^2 u^n = (u^{n+2/3} - u^n)/\Delta t \qquad (6.66)$$

$$z: \quad \Delta_x^2(u^{n+1/3} + u^n)/2 + \Delta_y^2(u^{n+2/3} + u^n)/2 + \Delta_z^2(u^{n+1} + u^n)/2 = (u^{n+1} - u^n)/\Delta t \qquad (6.67)$$

Eqs. 6.65–6.67 lead to the same overall equation given in Eq. 6.64; thus, the truncation error is the same as Brian's method. The only difference between the two is that the computed intermediate values are different.

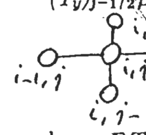
6.8 Band Matrix Problems

If the pressure in a reservoir remains essentially constant over a long period of time then the right-hand side of Eq. 6.9 is zero. From a practical point of view, such *steady-state* conditions exist only when the reservoir has been undisturbed (i.e., prior to drilling), at abandonment, under full pressure maintenance, or when the single-phase fluid is incompressible ($c = 0$). The resultant equation is then *elliptic* (see Appendix A.5.2). Any of the iterative ADI schemes discussed thus far can be applied to such a system. Consequently, a simulator that solves the 2 or 3-D parabolic problem given by Eqs. 6.20–6.21 and Eqs. 6.50–6.52 can equally well handle the companion elliptic problem that one obtains by setting $c = 0$. This is readily accomplished because c is usually specified as input data to the program. In so doing, the notion of taking a step in time vanishes, and we view the iterations as steps between different steady-state levels.

We now draw attention to direct methods of solution that depart from alternating direction schemes. This involves examining the structure of the

matrices involved and developing an algorithm that capitalizes on their structures. Consider Eq. 6.12 in its expanded form

$$-(T_y)_{j-1/2}p_{j-1} - (T_x)_{i-1/2}p_{i-1} + \left(\Sigma T + \frac{\alpha}{\Delta t}\right)p_{ij} - (T_x)_{i+1/2}p_{i+1} - (T_y)_{j+1/2}p_{j+1}$$

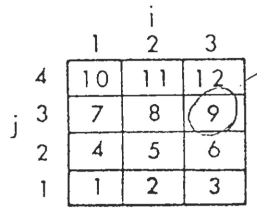


$$= \frac{\alpha}{\Delta t}p_{ij}^n + q_{ij}^n \equiv \tilde{q}_{ij}. \qquad (6.68)$$

where $\Sigma T = (T_x)_{i-1/2} + (T_x)_{i+1/2} + (T_y)_{j-1/2} + (T_y)_{j+1/2}$ as before.

In Eq. 6.68 we show only the fastest moving indices and suppress the superscript, $n + 1$, on the left-hand side. To examine the matrix structure arising from Eq. 6.68, suppose $N_x = 3$ and $N_y = 4$ as shown in Fig. 6.4.

توضیح ترتیب صفر



بطور نمونه

$m = i + (j-1)N_x \rightarrow$ مثال

$9 = 3 + (3-1)\times 3$

$11 = 2 + (4-1)\times 3$

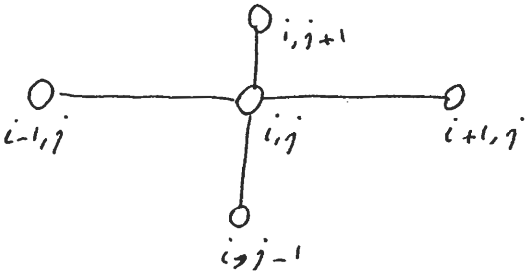
Fig. 6.4 Rectangular 2-D Reservoir.

The numbers around the periphery of the larger rectangle are the *(i,j)* coordinate indices. The grid blocks are sequentially numbered beginning at the bottom progressing upward from left to right. We call these numbers the *block indices* and the ordering *normal grid ordering.* The block indices, m, are related to the coordinate indices by the relationship $m = i + (j-1)N_x$. For example, if $i = 2$ and $j = 2$ then $m = 5$. Notice also that the left-hand side of Eq. 6.68 involves the blocks immediately to the right and left and above and below the *(i,j)* block. If the block index corresponding to the *(i,j)* block is m, then the blocks to the left, right, above and below will have indices $m - 1$, $m + 1$, $m + N_x$, $m - N_x$, respectively. This is readily seen by referring to Fig. 6.4. Consequently, we can express Eq. 6.68 in terms of the single block index, i.e.,

$$B_m p_{m-N_x} + D_m p_{m-1} + E_m p_m + F_m p_{m+1} + H_m p_{m+N_x} = \tilde{q}_m, \quad m = 1, 2, \ldots, N \qquad (6.69)$$

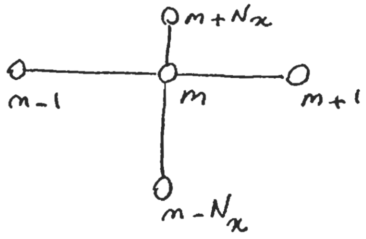
where $N = N_x N_y$, $B_m = -(T_y)_{j-1/2}$, $D_m = -(T_x)_{i-1/2}$, $E_m = \Sigma T$, $F_m = -(T_x)_{i+1/2}$, and $H_m = -(T_y)_{j+1/2}$. When m takes on values equal to the block indices on the boundary, then some of the terms in Eq. 6.69 refer to pressures in

— ارتباط سلول‌های محاسباتی معادله 6.68 با اندیس گذاری دو بعدی (i و j)
(i از چپ به راست افزایش می‌یابد و j از پایین به بالا)



— ارتباط سلول‌های محاسباتی معادله 6.68 با اندیس گذاری یک بعدی (m)
(مناسب برای پیاده سازی در حافظه خطی کامپیوتر، حافظه یک‌بعدی)

$m = i + (j-1)N_x$



blocks not included in the system. In these cases, the boundary conditions will be such that these terms can be deleted from the system of equations. In fact, the coefficients multiplying those terms will usually be zero. For purposes of illustration, we assume the latter is true. Consequently, for the 3×4 problem, Eq. 6.69 generates the following system of equations:

$$
\begin{array}{lllllllllllll}
E_1p_1 & +F_1p_2 & & +H_1p_4 & & & & & & & & & = \tilde{q}_1 \\
D_2p_1 & +E_2p_2 & +F_2p_3 & & +H_2p_5 & & & & & & & & = \tilde{q}_2 \\
 & +D_3p_2 & +E_3p_3 & & & +H_3p_6 & & & & & & & = \tilde{q}_3 \\
B_4p_1 & & & +E_4p_4 & +F_4p_5 & & +H_4p_7 & & & & & & = \tilde{q}_4 \\
 & B_5p_2 & & +D_5p_4 & +E_5p_5 & +F_5p_6 & & +H_5p_8 & & & & & = \tilde{q}_5 \\
 & & B_6p_3 & & +D_6p_5 & +E_6p_6 & & & +H_6p_9 & & & & = \tilde{q}_6 \\
 & & & B_7p_4 & & & +E_7p_7 & +F_7p_8 & & +H_7p_{10} & & & = \tilde{q}_7 \\
 & & & & B_8p_5 & & +D_8p_7 & +E_8p_8 & +F_8p_9 & & +H_8p_{11} & & = \tilde{q}_8 \\
 & & & & & B_9p_6 & & +D_9p_8 & +E_9p_9 & & & +H_9p_{12} & = \tilde{q}_9 \\
 & & & & & & B_{10}p_7 & & & +E_{10}p_{10} & +F_{10}p_{11} & & = \tilde{q}_{10} \\
 & & & & & & & B_{11}p_8 & & +D_{11}p_{10} & +E_{11}p_{11} & +F_{11}p_{12} & = \tilde{q}_{11} \\
 & & & & & & & & B_{12}p_9 & & +D_{12}p_{11} & +E_{12}p_{12} & = \tilde{q}_{12}
\end{array}
\quad \ldots \quad (6.70)
$$

In matrix form this becomes $\mathbf{Ap} = \mathbf{q}$ where A is of the form shown in Fig. 6.5(a). If this had been a 3-D problem, the structure would be the one shown in Fig. 6.5(b). The banded structures give rise to the name *band matrices*. The nonzero elements occur only on the solid parallel diagonals shown in Figs. 6.5(a) and 6.5(b). All other elements are zero. If the order of the matrix is large, the ratio of zeros to nonzeros is high, and the matrix is said to be *sparse*. Those one encounters in reservoir simulation are frequently sparse. Note in Figs. 6.5(a) and 6.5(b) that all the elements outside

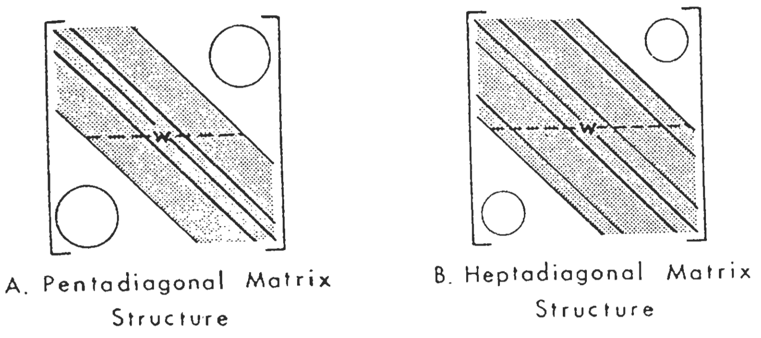

W
W
A. Pentadiagonal Matrix Structure
B. Heptadiagonal Matrix Structure

Fig. 6.5 Matrix Structures for 2-D and 3-D Problems.

of a band of width w centered on the main diagonal are zero. For our example 3×4 problem, the band width is $w = 7$. For normal grid ordering, the band width is equal to $2N_x + 1$ in 2-D and $2N_xN_y + 1$ in 3-D (taking the z-direction last). The amount of work involved in treating band matrix problems is roughly proportional to the square of the band width. Consequently, N_x and N_y should correspond to the shortest sides to minimize the band width.

6.8.1 Determination of Matrix Structures

To determine the matrix structure for a reservoir problem, we need only know the dimensionality of the reservoir. Consider, for example, a

توضیح شیب صفحه

Row Numbers	Column Numbers 1	2	3	4	5
1	x	x			
2	x	x	x		
3		x	x	x	
4			x	x	x
5				x	x

Fig. 6.6 Matrix Structure Setup.

1-D reservoir problem consisting of a collection of contiguous grid blocks. Suppose there are 5 blocks numbered 1 through 5. The matrix structure for flow through this reservoir is obtained as follows: Using a piece of gridded paper (graph paper is satisfactory), number 5 columns across the sheet and 5 rows down such that the numbers are equally spaced. This defines 25 row-column locations in all as depicted in Fig. 6.6.

Each grid block in the reservoir represents a unique row in the matrix. Furthermore, flow can potentially occur in each block to one or more blocks, through the faces between them. This determines the column locations where the nonzero elements occur in each row. For example, block 1 can support flow, but only to block 2. Thus for the first row in Fig. 6.6, we insert x's denoting nonzero elements in the first and second columns. Similarly, block 2 supports flow, possibly to 1 or to 3. Thus, in the second row above, we insert x's in columns 1, 2 and 3. This procedure is continued for all 5 blocks, and a tridiagonal structure emerges. Extending this idea to the 2-D problem in Fig. 6.4 gives the structure shown in Fig. 6.7.

← برپا کردن یا مشخصه سازی ماتریس : <u>برای حالت یک‌بعدی</u> : اگر فیزیک مخزن را به صورت مثلاً 5 بلوک زیر در نظر بگیریم:

1	2	3	4	5

$\xrightarrow{x}$

آنگاه ماتریس ضرایب به صورت 5×5 ظاهر می‌شود، ابتدا عناصر قطر اصلی را ضربدر بزنید، یعنی M_{11} ، M_{22} ، M_{33} و M_{44} و M_{55} :

$$M=\begin{bmatrix} \times & \cdot & \cdot & \cdot & \cdot \\ \cdot & \times & \cdot & \cdot & \cdot \\ \cdot & \cdot & \times & \cdot & \cdot \\ \cdot & \cdot & \cdot & \times & \cdot \\ \cdot & \cdot & \cdot & \cdot & \times \end{bmatrix}$$

سپس به فیزیک مخزن توجه کرده و هر بلوک مخزن را متناظر با هر سطر ماتریس M قرار دهید. حال بطور فیزیکی فرض کنید از چپ به راست flow داریم، از نقطه نظر سیال می‌تواند از بلوک 1 مخزن به بلوک 2 مخزن جابجا شود، پس اگر در M_{11} ایستاده باشیم (سطر اول یا بلوک 1 مخزن) فقط به سطر دوم، جریان داریم، یعنی M_{21}، پس ماتریس M به شکل زیر می‌شود:

$$M=\begin{bmatrix} \textcircled{x} & \cdot & \cdot & \cdot & \cdot \\ \times & \times & \cdot & \cdot & \cdot \\ \cdot & \cdot & \times & \cdot & \cdot \\ \cdot & \cdot & \cdot & \times & \cdot \\ \cdot & \cdot & \cdot & \cdot & \times \end{bmatrix}$$

برای بلوک 2 مخزن، می‌بینیم که جریان از بلوک 2 به بلوک‌های 1 و 3 مخزن داریم، پس اگر در M_{22} ایستاده باشیم، آنگاه فقط می‌توانیم یکبار به سمت بالا (به سمت سطر اول) و یکبار به سمت پایین (به سمت سطر سوم) برویم، پس M به شکل زیر در می‌آید:

$$M=\begin{bmatrix} \times & \times & \cdot & \cdot & \cdot \\ \times & \textcircled{x} & \cdot & \cdot & \cdot \\ \cdot & \times & \times & \cdot & \cdot \\ \cdot & \cdot & \cdot & \times & \cdot \\ \cdot & \cdot & \cdot & \cdot & \times \end{bmatrix}$$

به همین ترتیب ادامه می‌دهیم، یعنی روی M_{33} ایستاده، به طرف بالا و پایین حرکت می‌کنیم و ضربدر می‌زنیم، سپس روی M_{44} (حرکت به سمت بالا و پایین) و سپس روی M_{55} (حرکت فقط به سمت بالا)، نهایتاً، M به شکل زیر می‌شود:

$$M=\begin{bmatrix} \times & \times & \cdot & \cdot & \cdot \\ \times & \times & \times & \cdot & \cdot \\ \cdot & \times & \times & \times & \cdot \\ \cdot & \cdot & \times & \times & \times \\ \cdot & \cdot & \cdot & \times & \times \end{bmatrix}$$

← برپا کردن ماتریس برای حالت دوبعدی:

اگر فیزیک مخزن را به صورت مثلاً شکل زیر در نظر بگیرید: (3×4 یا 12 بلوک مخزن)

10	11	12
7	8	9
4	5	6
1	2	3

$\xrightarrow{x}$ $\uparrow y$

آنگاه ماتریس ضرایب به صورت 12×12 ظاهر شده و مثل حالت یک‌بعدی، ابتدا عناصر روی قطر را ضربدر بزنید (یعنی غیرصفر هستند). از M_{11} که متناظر با بلوک $\boxed{1}$ مخزن است شروع کنید. با توجه به فیزیک مخزن، بلوک $\boxed{1}$ ، فقط به بلوک $\boxed{2}$ و $\boxed{4}$ جریان دارد، پس (به سمت سطرهای دوم و چهارم حرکت می‌کنیم و ضربدر می‌زنیم:

$$M=\begin{bmatrix}
\textcircled{x} & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\times & \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\times & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \times & \cdot & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \times & \cdot & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \times & \cdot \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \times
\end{bmatrix}$$

سپس به المان M_{22} (یعنی بلوک $\boxed{2}$) می‌آییم و می‌بینیم جریان از بلوک $\boxed{2}$ می‌تواند

[illegible] $\boxed{3}$ [illegible] $\boxed{1}$ [illegible]

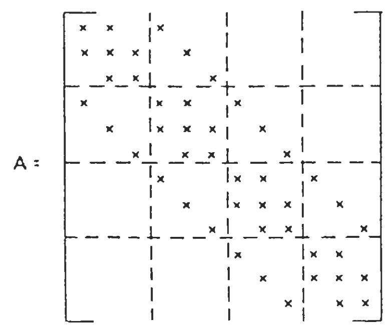


Fig. 6.7 Matrix Structure for a 3 × 4 × 1 Areal Reservoir Problem.

6.68 معادل

For single-phase flow, the off-diagonal elements in the matrix consist of transmissibilities at the block interfaces. Each diagonal element consists of the sums of the transmissibilities about the faces of the grid block corresponding to the row, plus some possible additional terms. The additional terms come from a well within the block (if any) and terms brought over from the right-hand sides of the finite difference equations. The net effect is that the matrix will usually be strictly diagonally dominant. (توضیح ردیف صفحه)

A direct solution technique for band matrices can be obtained by employing LU decomposition (توضیح ردیف صفحه ۱۰۵ آمده است). This procedure involves three steps: (1) factorization, (2) forward solution, (3) backward solution. Define $\hat{w} = (w-1)/2$ and let

$$L_1(i) \equiv \max\,(1,\ i-\hat{w})$$

$$L_2(i) \equiv \min\,(N,\ i+\hat{w})$$

$$L_3(j) \equiv \max\,(1,\ j-\hat{w})$$

The Crout algorithm for LU decomposition of a band matrix is

$$l_{ij} = a_{ij} - \sum_{k=L_1}^{j-1} l_{ik}u_{kj},\quad j = L_1,\ L_1+1,\ \ldots,\ i \qquad (6.71)$$

$$u_{ij} = \Big(a_{ij} - \sum_{k=L_3}^{i-1} l_{ik}u_{kj}\Big)/l_{ii},\quad j = i+1,\ \ldots,\ L_2(i) \qquad (6.72)$$

$$y_i = \Big(b_i - \sum_{k=L_1}^{i-1} l_{ik}y_k\Big)/l_{ii},\quad i = 1,\ 2\ \ldots,\ N \qquad (6.73)$$

$$x_i = y_i - \sum_{k=i+1}^{L_2} u_{ik}x_k,\quad i = N,\ N-1,\ \ldots,\ 1 \qquad (6.74)$$

Eqs. 6.71 and 6.72 are the factorization steps, Eq. 6.73 is the forward solution and Eq. 6.74 is the back solution.

This algorithm takes advantage of the banded structure of the matrix since it eliminates operations on the zero elements outside the band. However, there are still some wasted operations on the zeros within the band. By wasted operations, we mean addition or multiplication by zero. When the bandwidth is large, the algorithm becomes less efficient. In such cases, special sparse matrix techniques can be employed which involve operations only on the nonzero elements. If there are no zeros in the band, as in a tridiagonal matrix, the band algorithm cannot be surpassed in efficiency. Indeed, the Thomas tridiagonal algorithm cited in chapter 5 is a special case of the band algorithm and can be easily derived from it.

6.9 Ordering Schemes and Sparse Matrix Methods

Recall that in Gaussian elimination, we create an upper triangular matrix U from the original matrix A. This involves eliminating nonzeros below the main diagonal, and processing rows 2, 3, . . . , N one at a time. In so doing, we sometimes create nonzeros, below the current row being processed, where zero elements originally existed. These are called "fill" elements which must be stored and eventually eliminated themselves. By reordering the equations, we sometimes can reduce both storage and work requirements. This is accomplished by exploiting the matrix sparsity and reducing the number of fill elements created. For example, suppose we have a matrix with the structure depicted in Fig. 6.8 where x represents a nonzero element. If we upper triangulate this matrix as if it was full, we perform $(n^3 + 3n^2 - n)/3 = 36$ arithmetic operations (divides and multiplies). However, observe that all elements below the main diagonal, with the exception of a_{41}, are already zero. Thus, if we skip to the last equation and eliminate a_{41} only, we achieve the upper triangularization in 12 operations. Furthermore, this is accomplished without creating fill. Essentially, this amounts to reordering the rows of the matrix. For example, if the computer code recognizes that once the last row is processed, the elimination is complete, and it processes the last row first, it will stop thereafter, avoiding unnecessary work.

Obviously, developing computer code that can look ahead, assess where the nonzeros and potential fill elements are located, and then arrive at

تعریف ماتریس قطری غالب:

اگر ماتریس نمونه‌ای مثل A با المان‌های a_{ij}، دارای این خاصیت باشد که عناصر روی قطر از نظر اندازه (یعنی قدرمطلق) بزرگتر از مجموع عناصر دیگر روی همان سطر باشد، آنگاه ماتریس مزبور، "قطری غالب" است. به زبان ریاضی، اگر شرط زیر برقرار باشد آنگاه ماتریس قطری غالب است:

$$|a_{ii}| \geqslant \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}| \quad , \quad 1 \leqslant i \leqslant n$$

* نکته: اگر فقط علامت $>$ برقرار باشد آنگاه ماتریس اکیداً قطری غالب است.

* نکته: در تکرار اول، شاید ماتریس قطری غالب نباشد ولی با تعویض سطرها شاید بشود، دانیم.

فاکتوریزاسیون LU (Lower - Upper Factorization)

یک راه حل دستگاه جبری $A\underline{x} = \underline{b}$ استفاده از فاکتوریزاسیون LU یعنی تبدیل A به حاصلضرب دو ماتریس بالامثلثی واحد و پایین مثلثی می‌باشد. از این فاکتوریزاسیون نه تنها برای حل دستگاه، بلکه برای محاسبه دترمینان و هم چنین ماتریس وارون، بدون نیاز به معکوس‌گیری می‌باشد. برای حل دستگاه، معمولاً سه مرحله زیر را دنبال می‌کنند:

الف) عمل فاکتوریزاسیون (Factorization)

ب) حل صریح (Forward Solution)

ج) جایگزینی تراجعی (Backward Substitution) یا (B.S.)

تعاریف:

دستگاه	:	$A\underline{x} = \underline{b}$
ماتریس ضرایب	:	$A \triangleq [a_{ij}] \quad i,j=1,2,\dots,n$
ماتریس پایین مثلثی	:	$L \triangleq [l_{ij}] \ , \ l_{ij}=0 \text{ for } i<j$
ماتریس بالامثلثی واحد	:	$U \triangleq [u_{ij}] \ , \ \begin{cases} u_{ij}=0 & \text{for } i>j \\ u_{ij}=1 & \text{for } i=j \end{cases}$
بردار مجهول	:	$\underline{x}$
بردار میانی	:	$y \triangleq U.\underline{x}$

ادامه ←

ادامه ← سیر حل دستگاه به صورت زیر است

$$A\underline{x}=\underline{b} \xrightarrow[A=LU\,(\text{الف})]{\text{فاکتوریزاسیون}} LU\underline{x}=\underline{b} \quad \text{یا} \quad L\underline{y}=b \xrightarrow[(\text{ب})]{\text{حل صریح}} \underline{y} \xrightarrow[(\text{ج})]{B.S.} \underline{x}$$

الف) فاکتوریزاسیون

$$A=LU \xrightarrow{\text{با بسط سطری}} a_{ij} = \sum_{k=1}^{n} l_{ik}u_{kj} \ , \ j=1,\dots,n$$

$$= \sum_{k=1}^{j-1} l_{ik}u_{kj} + l_{ij}u_{jj} + \cancelto{0}{\sum_{k=j+1}^{n} l_{ik}u_{kj}} \ , \ j=1,\dots,n$$

$$\rightarrow \boxed{l_{ij} = a_{ij} - \sum l_{ik}u_{kj}} \quad \text{ستون عناصر } L$$

$$\left(\begin{matrix}\text{با بسط ستونی} \\ i=1,2,\dots,n\end{matrix}\right) a_{ij} = \sum_{k=1}^{i-1} l_{ik}u_{kj} + l_{ii}u_{ij} + \cancelto{0}{\sum_{k=i+1}^{n} l_{ik}u_{ki}}$$

$$\rightarrow \boxed{u_{ij} = \left(a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}\right)/l_{ii}} \quad \text{سطر عناصر } U$$

ب) حل صریح

$$L\underline{y}=\underline{b} \rightarrow b_i = \sum_{k=1}^{n} l_{ik}y_k \ , \ i=1,\dots,n$$

$$= \sum_{k=1}^{i-1} l_{ik}y_k + l_{ii}y_i + \cancelto{0}{\sum_{k=i+1}^{n} l_{ik}y_k}$$

$$\rightarrow \boxed{y_i = \left(b_i - \sum_{k=1}^{i-1} l_{ik}y_k\right)/l_{ii}}$$

ج) جایگزینی تراجعی (B.S.)

$$U\underline{x}=\underline{y} \rightarrow$$

$$y_i = \cancelto{0}{\sum_{k=1}^{i-1} u_{ik}x_k} + u_{ii}x_i + \sum_{k=i+1}^{n} u_{ik}x_k$$

$$\rightarrow \boxed{x_i = y_i - \sum_{k=i+1}^{n} u_{ik}x_k \ , \ i=n-1,n-2,\dots,1}$$

$$A = \begin{bmatrix} x & o & o & x \\ o & x & o & x \\ o & o & x & x \\ x & o & o & x \end{bmatrix}$$

Fig. 6.8 Hypothetical Matrix Structure.

the best reordering to minimize work and storage is rather tricky. However, it can be done. Software packages that do this are called *sparse matrix routines.*[15] Their common characteristics are reordering of the matrix elements, followed by operations on the nonzeros only. Such routines frequently employ **LU** decomposition for solving the matrix problem, rather than Gaussian elimination, because the former procedure preserves the original sparsity of the matrix as much as possible where Gaussian elimination does not.[16]

Reordering the matrix elements is equivalent to renumbering the grid blocks of the reservoir. For example, consider a reservoir that is shaded checkerboard fashion as in Fig. 6.9. If we number the dark blocks first and then the white blocks, from top to bottom, beginning from the left, we get a substantially different matrix structure. A typical matrix is shown for alternate diagonal ordering of a 30 × 30 system in Fig. 6.10. This is also called D-4 or white-black ordering. This ordering scheme was first employed in the electrical power industry for network problems[17] and subsequently discussed by Price and Coats in the petroleum literature.[18] Note that it's necessary to triangularize only the lower half of the matrix. Consequently, the work† and storage is cut by a factor of at least two during Gaussian elimination. When the triangularization is completed, the matrix U has the form shown in Fig. 6.11 where the small circles are fill elements.

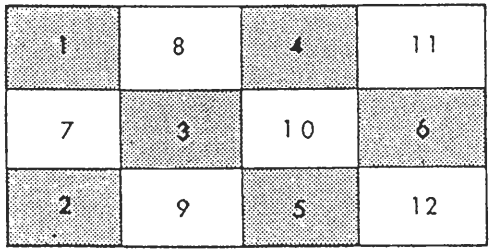


Fig. 6.9 Alternate Diagonal Ordering.

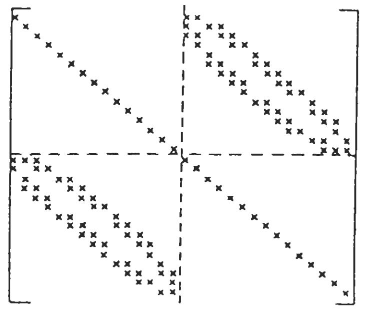


Fig. 6.10 Matrix Structure for Alternate Diagonal Ordering.

Some simulators offer the option of using alternate diagonal ordering in conjunction with Gaussian elimination. For certain problems, this can produce substantial savings in cost and computer storage. For example, for a 2-D areal problem we have $W_1 \simeq IJ^3$ and $S_1 \simeq IJ^2$ for large I and J $(J < I,\ I = N_x,\ J = N_y)$, where W_1 and S_1 are work and storage, respectively for normal grid ordering. For D-4 ordering,

$$W_4 \simeq IJ^3/2 - J^4/4 \qquad (6.75)$$

$$S_4 \simeq IJ^2/2 - J^3/6 \qquad (6.76)$$

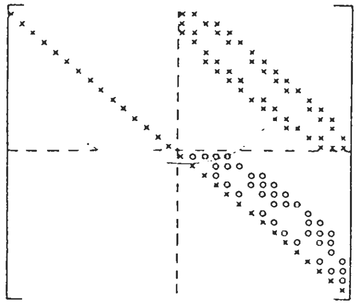
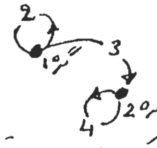


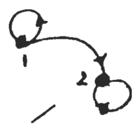
Fig. 6.11 Matrix U from Alternate Diagonal Ordering.

† We use "work" in the sense of operations a computer must perform. This includes adds, multiplies, divides, fetches and stores.

گراف یک ماتریس، مفاهیم درخت و حلقه:

فرض کنید $A_{n\times n}$ یک ماتریس مربعی باشد. آنگاه ما به ازای ماتریس A یک گراف با n گره و چند یال (vertex) یا ضلع (edge) مثل G(A) یا G(X,E) تعریف می‌کنیم، به طوری که X مجموعه گره‌ها و E مجموعه یال‌های گراف می‌باشند. اگر برای تمام المان‌های غیرصفر از ماتریس A (مثل a_{ij}) از گره i ام ($P_i \triangleq$) به گره j ام ($P_j \triangleq$) وصل کنیم به یک دیاگراف (Directed Graph یا Digraph) می‌رسیم. اگر برای هر زوج مرتب از گره‌ها، حداقل یک مسیر اتصال وجود داشته باشد آنگاه می‌گوییم دیاگراف متصل (Connected) است. به طور مثال، ماتریس $A=\begin{bmatrix}2&3\\0&4\end{bmatrix}$ را در نظر بگیرید. گراف A، یک دیاگراف غیرمتصل است، چون المان a_{21} ($E_{②\to①}$) صفر است یا وجود ندارد.

$A=\begin{bmatrix}2&3\\0&4\end{bmatrix} \to G(A):$ 

یا 

نکته: اگر دیاگراف یک ماتریس، متصل باشد، آنگاه غیرقابل تقلیل پذیر است.

اگر ترتیب یا مسیر برای ما مهم نباشد، آنگاه یک گراف داریم که جهت برای ما مهم نیست، یعنی الزاماً دیاگراف نداریم و می‌گوییم Undirected Graph.

مفهوم زیرگراف (Subgraph): اگر یک گراف داشته باشیم مثل G با G(X,E) به طوری که X و E، مجموعه گره‌ها و یال‌ها باشند، مشخصه سازی کنیم، آنگاه G(X',E') یک زیرگراف از G است به طوری که X' و E' زیرمجموعه‌هایی از X و E باشند.

نکته: هر دیاگراف متصل، دقیقاً یک زیرگراف متصل دارد.

تعریف حلقه: حلقه یک زیرگراف یا گراف است که دارای مسیری باشد که از یک گره شروع کرده و نهایتاً به همان گره بتوانیم برسیم.

تعریف درخت: درخت یک گراف متصل است که دارای حلقه نباشد.

قضیه: ماتریس‌هایی که گراف آنها، درخت هستند، می‌توانند طوری مرتب شوند که در عملیات حذفی گوس، دارای عنصر fill نباشند.

مفهوم ماتریس‌های تقلیل پذیر

فرض کنید A یک ماتریس مربعی n×n باشد. ماتریس A تقلیل پذیر است اگر بتوان یک ماتریس جایگشتی (permutation) مثل P پیدا کرد به طوری که

$$PAP^T=\begin{bmatrix}A_{11}&A_{12}\\0&A_{22}\end{bmatrix}$$

به طوری که

$A_{11} \triangleq r\times r$ submatrix , $r<n$

$A_{22} \triangleq (n-r)\times(n-r)$ submatrix

$0 \triangleq (n-r)\times r$ null submatrix

$A_{12} \triangleq r\times(n-r)$ submatrix

مفهوم ماتریس جایگشتی (permutation Matrix) (کاربرد در Ordering)

به ماتریسی که در هر سطر یا ستون آن، فقط یک عنصر آن، 1 باشد و بقیه صفر، می‌گوییم ماتریس جایگشتی. مثلاً ماتریس جایگشتی $P=\begin{bmatrix}0&1\\1&0\end{bmatrix}$ را در نظر بگیرید. اگر این ماتریس را در ماتریس نمونه A ضرب کنیم، حاصلضرب همان A می‌باشد ولی اندیس سطرهایش عوض شده‌اند!

$$A=\begin{bmatrix}a_{11}&a_{12}\\a_{21}&a_{22}\end{bmatrix} \to PA=\begin{bmatrix}0&1\\1&0\end{bmatrix}\begin{bmatrix}a_{11}&a_{21}\\a_{21}&a_{22}\end{bmatrix}$$

$$\to \quad PA=\begin{bmatrix}a_{21}&a_{22}\\a_{11}&a_{12}\end{bmatrix}$$

نکته: به ماتریس جایگشتی زیر دقت کنید، اگر هر 1، معادل یک وزیر در شطرنج باشد، آنگاه با آرایش زیر، هیچ وقت وزیرها همدیگر را نمی‌زنند! جالبه.

$$P=\begin{bmatrix}0&0&0&0&0&1&0&0\\0&1&0&0&0&0&0&0\\0&0&0&0&0&0&1&0\\1&0&0&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&0&0&0&1\\0&0&0&0&1&0&0&0\\0&0&1&0&0&0&0&0\end{bmatrix}$$

For 2-D problems,

$$W_4/W_1 = (2\ell - 1)/4\ell \tag{6.77}$$

where $\ell = I/J$. Thus, for highly elongated rectangles $(I > J)$, D-4 is twice as fast as standard Gaussian elimination and four times faster when $\ell = 1$.

In [illegible], we provide some basic definitions regarding the graph of a matrix, **A**. The notation and theory of graphs is especially useful in treating sparse matrices.[19] With regard to the latter, our concern is primarily with undirected graphs. It would be nice if one could find an ordering that produced a matrix whose graph is a tree since, in this case, Gaussian elimination produces no fill. However, for the banded matrices encountered in reservoir simulation, this does not appear possible. The best one can hope for is an optimal or near-optimal ordering that minimizes the number of fill elements. To date such approaches have not been fully investigated relative to reservoir simulation problems. A number of near-optimal ordering algorithms have been proposed for electrical networks.[20] They are not guaranteed, however, to produce the minimum in fills, but in many practical cases, they do so.

Given a matrix problem $\mathbf{Ax} = \mathbf{b}$, elimination of unknowns is equivalent to eliminating nodes in the graph G of A to produce a subgraph G'. The subgraph is derived from G as follows:

(1) For every pair of edges (a,b) and (a,c) in G incident at node a, construct a new edge joining b and c. If this edge is connected in parallel with an existing one, represent the two as a single edge.
(2) Delete node a and all edges incident at node a. The resulting graph is G'.
(3) If in step 1 the new edge is not a parallel edge with an existing one, then a fill element will be created in the upper triangular matrix **U**.†

To illustrate, consider the 6 × 6 matrix in Fig. 6.12 where, again, the x's represent nonzeros. The graph of A is displayed in Fig. 6.13 where we omit the loops corresponding to the diagonal elements. If the elimination of unknowns is in the order, $x_1, x_2, \ldots$, then we get the subgraphs shown in Fig. 6.14 for the first four eliminations. Obviously no fill will occur beyond this point. The total fill is 4 (2 each in the **L** and **U** matrices). The idea in optimal ordering schemes from the graph theory point of view, is to seek

† The **U**-matrix created from **LU**-decomposition is identical with that obtained by Gaussian elimination. Consequently the fill in **U** will be the same for both methods. Therefore, it suffices to determine the fill in **U** only, and double it to get the fill in **L** and **U**.

$$\mathbf{A} = \begin{bmatrix} x & o & o & x & o & x \\ x & x & o & x & o & o \\ o & o & x & x & o & x \\ x & o & o & x & x & o \\ o & x & o & x & x & o \\ x & o & x & x & o & x \end{bmatrix}$$

Fig. 6.12 Example 6 × 6 Matrix.

that ordering that produces a collection of subgraphs with the minimum number of new edges.

A typical near-optimal ordering scheme is Berry's.[21] He describes it in terms of matrix manipulations. Here we describe it in terms of equivalent matrix graph reductions of an n^{th} order matrix. Assume the *incidence matrix*

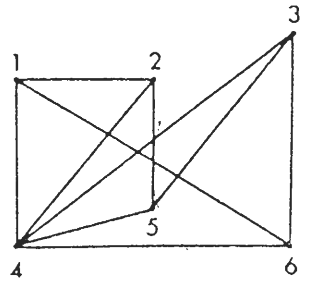
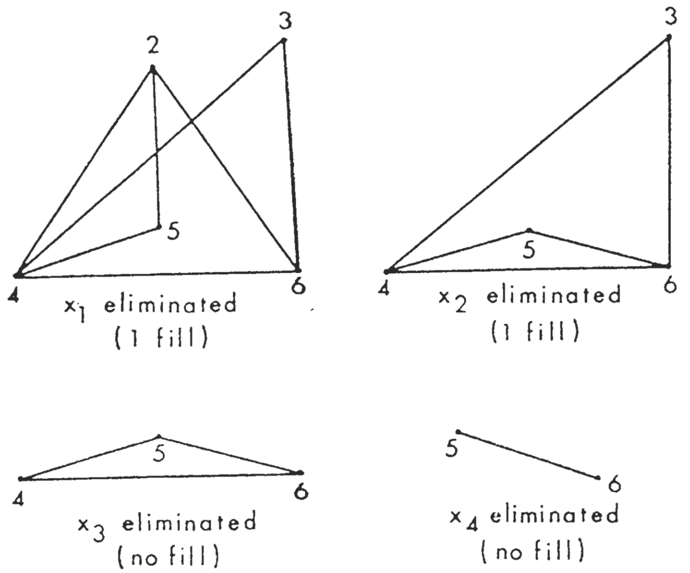


Fig. 6.13 Graph of Matrix A.

of A is symmetric,† $a_{ii} \neq 0$ for all i, and when LU factorization is performed, $\ell_{ii} \neq 0$ for all i. The algorithm is presented in flow chart form in Fig. 6.15. If we apply this to the graph in Fig. 6.13, we find it fails the first two tests; i.e., there are no nodes with only one incident edge, nor are there any, which when removed, yield a parallel edge with an existing edge. Consequently, we must seek a node yielding minimum fill. Removal of any one of nodes 1, 2, 3, 5, or 6 will yield one fill element (a tie). Removal

† The incidence matrix, **M** say, of **A** is one where $m_{ij} = 1$ if $a_{ij} \neq 0$ and $m_{ij} = 0$ if $a_{ij} = 0$.



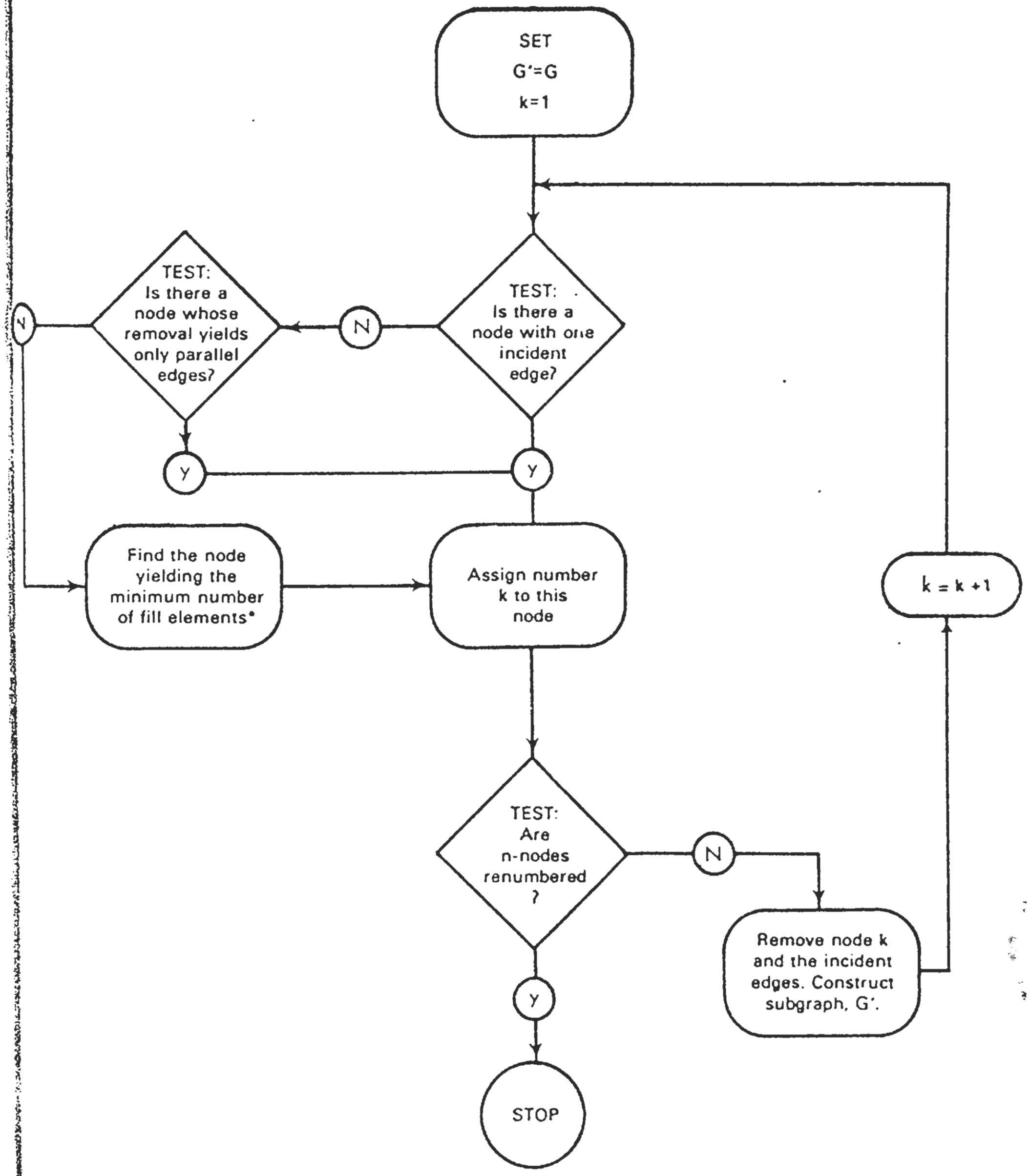
x_1 eliminated (1 fill)
x_2 eliminated (1 fill)
x_3 eliminated (no fill)
x_4 eliminated (no fill)

Fig. 6.14 Subgraphs of Matrix A.

of node 4 first (the worst choice) yields five fills. Each of nodes 1, 2, 3, 5, and 6 have three incident edges so it is immaterial which of these is selected as the first eliminant. Suppose node 1 is selected. The resultant subgraph is in the upper left-hand corner of Fig. 6.14. The first test in the flow chart is again applied and the subgraph fails; however, it does not fail the second test. Clearly, node 5 upon removal yields a parallel edge, hence it is renumbered as 2. Finally nodes 2, 3, and 4 are renumbered and removed. The entire process results in two fill elements after doubling. The renumbering is as follows: 1 → ①; 5 → ②; 2 → ③; 3 → ④; 4 → ⑤; and 6 → ⑥ where the circled values represent the new node orderings. Notice that 1 → ①; 5 → ②; 3 → ③; 2 → ④; 6 → ⑤; 4 → ⑥ also yields two fill elements.

6.10 Strongly Implicit Procedure

For reservoir problems requiring a large number of grid blocks, the storage requirements can become excessively high for direct solution methods. Even sparse matrix techniques can lose their utility since a substantial overhead is incurred in maintaining pointers for the original and created nonzeros.[16] Moreover, all advantages of sparse matrix routines are lost if they are programmed poorly. Under such circumstances, an alternative is to employ



SET G'=G k=1
TEST: Is there a node with one incident edge?
N
TEST: Is there a node whose removal yields only parallel edges?
Y
Y
N
Find the node yielding the minimum number of fill elements*
Assign number k to this node
TEST: Are n-nodes renumbered ?
N
Remove node k and the incident edges. Construct subgraph, G'.
k = k + 1
Y
STOP

* To achieve this, each node is examined and the fills are determined assuming that particular node only is eliminated. The node leading to the minimum number of fill

iterative techniques. Their principal advantage is that storage requirements are minimal. The main disadvantage is that some iterative techniques, under the best of circumstances, fail to converge. This occurs frequently with the IADI methods where the transmissibilities have a high contrast in one direction relative to the others. As an alternate iterative method Stone[22] introduced the Strongly Implicit Procedure (SIP) that readily handles many of the difficult problems of reservoir simulation. Furthermore, it has the advantage that iteration parameters can be more easily selected. It is equally applicable to parabolic and elliptic systems in both 2-D and 3-D. We briefly discuss the SIP algorithm here in the context of the 2-D parabolic problem given in Eq. 6.12.

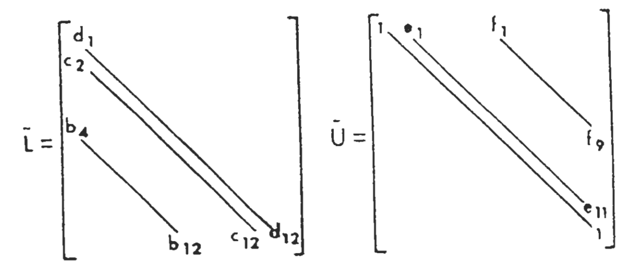


Fig. 6.16 $\tilde{L}$ and $\tilde{U}$ for a 3 × 4 problem.

Consider the problem $\mathbf{Ax} = \mathbf{b}$. The idea is to replace $\mathbf{A}$ with a matrix $\tilde{\mathbf{A}}$ that is sufficiently close to $\mathbf{A}$ in some sense, but yields a faster rate of convergence when used as an iteration matrix. Suppose we construct $\tilde{\mathbf{A}}$ and factor it into a lower triangular matrix $\tilde{\mathbf{L}}$, and a unit upper triangular matrix $\tilde{\mathbf{U}}$. In each of these factors there are at most three elements per row (for a 2-D problem). Since $\tilde{\mathbf{A}}$ is close to $\mathbf{A}$, factorization of $\tilde{\mathbf{A}}$ can be regarded as an approximate factorization of $\mathbf{A}$. The $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{U}}$ matrices are then used in a sequential forward and backward solution procedure. In this regard, the steps involved, i.e., factorization, forward solution, and back solution are essentially those in **LU** decomposition described previously.

The algebraic expansion of Eq. 6.12 is given in Eq. 6.69. In Fig. 6.16 we depict factors $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{U}}$ for the 3 × 4 problem considered in section 8. If we premultiply $\tilde{\mathbf{U}}$ by $\tilde{\mathbf{L}}$ then $\tilde{\mathbf{A}}$ has the structure shown in Fig. 6.17. Compare $\mathbf{A}$ and $\tilde{\mathbf{A}}$ and note that two additional diagonals are introduced that fall just inside the $\tilde{B}$ and $\tilde{H}$ diagonals. We have the following relationships between the elements in $\tilde{\mathbf{A}}$ and those in $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{U}}$.

$$\tilde{B}_m = b_m \tag{6.78a}$$

$$\tilde{C}_m = b_m e_{m-N_x} \tag{6.78b}$$

$$\tilde{D}_m = c_m \tag{6.78c}$$

$$\tilde{E}_m = b_m f_{m-N_x} + c_m e_{m-1} + d_m \tag{6.78d}$$

$$\tilde{F}_m = d_m e_m \tag{6.78e}$$

$$\tilde{G}_m = c_m f_{m-1} \tag{6.78f}$$

$$\tilde{H}_m = d_m f_m \tag{6.78g}$$

These relationships must all be satisfied so that $\tilde{\mathbf{A}}$ is factorable. On the other hand, we have for every block m, the five unknowns b_m, c_m, d_m, e_m,

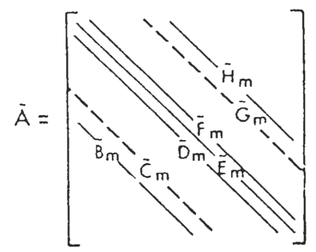


Fig. 6.17 Structure of $\tilde{A}$.

f_m. Thus, of the seven relationships in Eq. 6.78 five of the seven capitalized coefficients are arbitrary while the remaining two can be related to the other five using Eq. 6.78. Of course it is also necessary to relate the coefficients of $\tilde{\mathbf{A}}$ to $\mathbf{A}$. Stone made the following choices:

$$\tilde{B}_m = B_m - \sigma \tilde{C}_m \tag{6.79a}$$

$$\tilde{D}_m = D_m - \sigma \tilde{G}_m \tag{6.79b}$$

$$\tilde{E}_m = E_m + \sigma(\tilde{C}_m + \tilde{G}_m) \tag{6.79c}$$

$$\tilde{F}_m = F_m - \sigma \tilde{C}_m \tag{6.79d}$$

$$\tilde{H}_m = H_m - \sigma \tilde{G}_m \tag{6.79e}$$

where σ is an iteration parameter. These relationships lead to the matrix equation

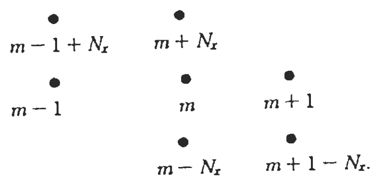
$$B_m p_{m-N_x} + \check{C}_m \bar{p} + D_m p_{m-1} + E_m p_m + F_m p_{m+1} + \breve{G}_m \hat{p} + H_m p_{m+N_x} = \tilde{q}_m \tag{6.80}$$

where

$$\bar{p} = p_{m+1-N_x} - \sigma(p_{m+1} + p_{m-N_x} - p_m), \tag{6.81}$$

$$\hat{p} = p_{m-1+N_x} - \sigma(p_{m-1} + p_{m+N_x} - p_m). \tag{6.82}$$

Note that Eq. 6.80 is identical with Eq. 6.69 except for the additional terms $\check{C}_m\bar{p}$ and $\breve{G}_m\hat{p}$ which introduce the additional points at $m + 1 - N_x$ and $m - 1 + N_x$. If $\bar{p}$ and $\hat{p}$ are sufficiently small, $\breve{A}$ will be close to A in some sense. This is especially true if the pressures are smooth over the grid. Note also that the computational star is now



If we combine Eqs. 6.79 and 6.78, we get the SIP factorization algorithm as follows:

$$b_m = \frac{B_m}{1 + \sigma e_{m-N_x}} \tag{6.83}$$

$$c_m = \frac{D_m}{1 + \sigma f_{m-1}} \tag{6.84}$$

$$d_m = E_m + \sigma(b_m e_{m-N_x} + c_m f_{m-1}) - b_m f_{m-N_x} - c_m e_{m-1} \tag{6.85}$$

$$e_m = \frac{F_m - \sigma b_m e_{m-N_x}}{d_m} \tag{6.86}$$

$$f_m = \frac{H_m - \sigma c_m f_{m-1}}{d_m} \tag{6.87}$$

These five equations can be solved sequentially for $m = 1, 2, \ldots, N$.

Once the factorization is completed using Eqs. 6.83–6.87, we use $\breve{A}$ as an iteration matrix, i.e., for the problem $\breve{A}\mathbf{p} = \mathbf{q}$, write

$$\breve{A}\mathbf{p}^{k+1} = \breve{A}\mathbf{p}^k - \omega(\mathbf{A}\mathbf{p}^k - \mathbf{q}). \tag{6.88}$$

Define a displacement vector $\mathbf{DP}^{k+1}$ and residual vector $\mathbf{R}^k$ by

$$\mathbf{DP}^{k+1} \equiv \mathbf{p}^{k+1} - \mathbf{p}^k$$

$$\mathbf{R}^k \equiv \mathbf{q}^k - \mathbf{A}\mathbf{p}^k$$

such that Eq. 6.88 becomes

$$\breve{A}\mathbf{DP}^{k+1} = \omega\mathbf{R}^k \tag{6.89}$$

with ω as an iteration parameter. Now Eq. 6.89 can be written as

$$\breve{L}\breve{U}\mathbf{DP}^{k+1} = \omega\mathbf{R}^k$$

or

$$\breve{L}\mathbf{v}^{k+1} = \omega\mathbf{R}^k, \tag{6.90}$$

and

$$\breve{U}\mathbf{DP}^{k+1} = \mathbf{v}^{k+1}. \tag{6.91}$$

Eqs. 6.90 and 6.91 are the forward and back solutions, respectively. Expanding each yields

$$b_m v_{m-N_x}^{k+1} + c_m v_{m-1}^{k+1} + d_m v_m^{k+1} = \omega R_m^k \tag{6.92}$$

$$DP_m^{k+1} + e_m DP_{m+1}^{k+1} + f_m DP_{m+N_x}^{k+1} = v_m^{k+1} \tag{6.93}$$

Eq. 6.92 is solved for $m = 1, 2, \ldots, N$ while in Eq. 6.93 $m = N - 1, N - 2, \ldots, 1$.

An alternative procedure to Stone's was proposed by Dupont, Kendall, and Rachford[23] where in Eqs. 6.81 and 6.82 they let $\bar{p} = p_{m+1-N_x} - p_m$ and $\hat{p} = p_{m-1+N_x} - p_m$ to reduce the effects of $\check{C}_m$ and $\breve{G}_m$. In addition, they replace E_m by $E_m + \beta$ where the latter is an iteration parameter. However, Stone's method appears to be superior. In the DKR method, varying ω can help to accelerate convergence, however, in Stone's method a fixed value of one is usually satisfactory. It is important to cycle through a set of σ-values. The minimum value is not critical and can be set to zero. The maximum value is the most sensitive parameter and can be estimated from $1 - \sigma_1$ where σ_1 is given in Eq. 6.24. As in ADI the parameters are geometrically spaced. For cases where the transmissibility ratios in Eq. 6.24 vary over the grid, local values of σ_{max} are computed and the maximum over the grid is used.

Although the algorithm presented here is for a 2-D single-phase flow problem, the technique has been extended by Weinstein, Stone, and Kwan[24] to 3-D problems and problems involving multiphase flow[25]. In all cases, except for 2-D single-phase flow in isotropic, homogeneous media, SIP achieved greater rates of convergence than iterative ADI. As the degree of the nonlinearities increases, the advantage of SIP over ADI appears to increase, and in many instances, no difficulty with SIP is experienced when ADI fails to converge at all.

6.11 Point Iterative Methods

For a given matrix problem, $\mathbf{Ax} = \mathbf{b}$, the matrix $\mathbf{A}$ can be expressed as the sum $\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U}$. $\mathbf{D}$ is a diagonal matrix containing the diagonal elements of $\mathbf{A}$, and $\mathbf{L}$ and $\mathbf{U}$ are strict lower and upper triangular matrices whose elements are the negatives of the elements of $\mathbf{A}$ below and above the main diagonal of $\mathbf{A}$. We write the problem as

$$\mathbf{Dx} = (\mathbf{L} + \mathbf{U})\mathbf{x} + \mathbf{b} \tag{6.94}$$

$$\mathbf{x}^{(k+1)} = \mathbf{M}_J \mathbf{x}^{(k)} + \mathbf{c} \tag{6.95}$$

where $\mathbf{M}_J \equiv \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})$, the *point Jacobi* iteration matrix and $\mathbf{c} \equiv \mathbf{D}^{-1}\mathbf{b}$. This procedure requires that one store all the components of the vector $\mathbf{x}^{(k)}$ while computing $\mathbf{x}^{(k+1)}$. An approach that does not require this is the *point Gauss-Seidel* method where the iteration matrix is $\mathbf{M}_{GS} \equiv (\mathbf{D} - \mathbf{L})^{-1}\mathbf{U}$ and $\mathbf{c} = (\mathbf{D} - \mathbf{L})^{-1}\mathbf{b}$. In reservoir simulation neither of these techniques are employed since the convergence rates are too slow. However, convergence can be accelerated using an amplification factor (or iteration parameter), ω. We define the displacement vector $\mathbf{d} = \mathbf{x}^{(k+1)} - \mathbf{x}^{(k)}$, i.e., it is the change in the solution vector over one iteration. The *point successive overrelaxation* (SOR) method is based on amplifying the displacement vectors obtained from the Gauss-Seidel procedure, i.e.,

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} + \omega \mathbf{d}_{GS}. \tag{6.96}$$

This leads to the following matrix form:

$$\mathbf{x}^{(k+1)} = (\mathbf{D} - \omega\mathbf{L})^{-1}\{(1 - \omega)\mathbf{D} + \omega\mathbf{U}\}\mathbf{x}^{(k)} + \omega(\mathbf{D} - \omega\mathbf{L})^{-1}\mathbf{b} \tag{6.97}$$

where the iteration matrix is $\mathbf{M}_\omega \equiv (\mathbf{D} - \omega\mathbf{L})^{-1}\{(1 - \omega)\mathbf{D} + \omega\mathbf{U}\}$ and the vector $\mathbf{c}$ is defined by the rightmost term in Eq. 6.97.

These techniques are referred to as "point" methods since the components in vector $\mathbf{x}$ are computed sequentially point-by-point. For example, the SOR method in terms of individual components becomes

$$a_{ii}x_i^{(k+1)} = a_{ii}(1 - \omega)x_i^{(k)} - \omega\left\{\sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k)} - b_i\right\} \tag{6.98}$$

for an n^{th} order matrix problem. A necessary and sufficient condition for convergence of the three techniques cited here is that the spectral radius of the iteration matrix be less than one.[26] Obviously, the spectral radius in the SOR method is determined by the choice of ω. It has been found that the SOR method will converge when $0 < \omega < 2$ and whenever $\mathbf{A}$ is a positive definite matrix.[27] This is frequently the case with reservoir simulation problems. The fastest convergence rate with SOR is obtained when an optimum iteration parameter, ω_o, is employed. By optimum we mean that choice of ω that produces the smallest possible value of $\rho(\mathbf{M}_\omega)$. An examination of convergence rates indicates that $1 < \omega_o < 2$, consequently, the choice of ω should be made within these bounds. Unfortunately, there is no easy way to compute ω_o for practical reservoir simulation problems. For single-phase flow in a 1 – D, homogeneous, isotropic reservoir ω_o is given by

$$\omega_o = \frac{2}{1 + \sqrt{1 - \sigma^2}} \tag{6.99}$$

where $\sigma \equiv \rho(\mathbf{M}_J)$. This can possibly be used for an initial estimate for multiphase, heterogeneous, anisotropic systems. Subsequent numerical experimentation with the simulator can then produce a better estimate. As an alternative to this, several mathematical procedures[28-31] are available for estimating ω_o.

Note that when $\omega = 1$, SOR becomes the Gauss-Seidel method. The convergence rate of the latter is approximately twice that of the Jacobi technique, while for a proper choice of ω, SOR is capable of an even faster rate.

6.12 Block Successive Overrelaxation

A point SOR method is readily extended to a *block* SOR technique by considering a partitioning of the problem $\mathbf{Ax} = \mathbf{b}$, e.g.,

$$\begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1n} \\ A_{21} & A_{22} & \cdots & A_{2n} \\ \vdots & \vdots & & \vdots \\ A_{n1} & A_{n2} & \cdots & A_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_n \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_n \end{bmatrix} \qquad (6.100)$$

where the A_{ij}'s are submatrices and the $\mathbf{x}_i$'s and $\mathbf{b}_i$'s contain subcomponents of $\mathbf{x}$ and $\mathbf{b}$. For example, the dashed lines on the pentadiagonal matrix shown in Fig. 6.7 define a partitioning into *block tridiagonal* form. The diagonal submatrices A_{ii} are themselves tridiagonal, while the off-diagonal submatrices are diagonal matrices. In this case, the partitions of the vectors $\mathbf{x}$ and $\mathbf{b}$ would each contain three components: $\mathbf{x}_1 = (x_1, x_2, x_3)^T$, $\mathbf{b}_1 = (b_1, b_2, b_3)^T$, $\mathbf{x}_2 = (x_4, x_5, x_6)^T$, $\mathbf{b}_2 = (b_4, b_5, b_6)^T$, etc. A block SOR technique utilizes the partitioned matrix form and replaces the scalar calculations in Eq. 6.98 with matrix calculations. Thus we have the analog of Eq. 6.98,

$$A_{ii}\mathbf{x}_i^{(k+1)} = A_{ii}(1-\omega)\mathbf{x}_i^{(k)} - \omega\left\{\sum_{j=1}^{i-1} A_{ij}\mathbf{x}_j^{(k+1)} + \sum_{j=i+1}^{n} A_{ij}\mathbf{x}_j^{(k)} - \mathbf{b}_i\right\} \quad i = 1, \ldots, n. \qquad (6.101)$$

A better computational algorithm is

$$A_{ii}\mathbf{y}_i^{(k+1)} = -\sum_{j=1}^{i-1} A_{ij}\mathbf{x}_j^{(k+1)} - \sum_{j=i+1}^{n} A_{ij}\mathbf{x}_j^{(k)} + \mathbf{b}_i, \; i = 1, 2, \ldots, n \qquad (6.102)$$

$$\mathbf{x}_i^{(k+1)} = \omega\left\{\mathbf{y}_i^{(k+1)} - \mathbf{x}_i^{(k)}\right\} + \mathbf{x}_i^{(k)}, \; i = 1, 2, \ldots, n \qquad (6.103)$$

where $\mathbf{y}_i^{(k+1)}$ is an intermediate vector. For the matrix displayed in Fig. 6.7, the intermediate vector can be obtained by the Thomas tridiagonal algorithm. Observe that each application of Eqs. 6.102 and 6.103 for $i =$ 1, 2, 3 and 4 yields one line of unknowns for the rectangular reservoir in Fig. 6.4. Thus this approach is called *line successive overrelaxation* (LSOR). If we partition the matrix about the central lines of symmetry in Fig. 6.7 such that $\mathbf{A} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ then the diagonal submatrices are pentadiagonal. The algorithm in Eqs. 6.102–6.103 is still applicable with the exception that the matrix problem in Eq. 6.102 is solved by the band algorithm (or possibly a sparse matrix technique for larger problems). Since this partitioning yields a simultaneous solution for the unknowns in two rows of grid blocks, it is called *two-line successive overrelaxation* (2LSOR).

The reason for passing from point to block SOR methods is that convergence rates are significantly better with the latter. Moreover, block methods generally incur less round-off error and can be performed in such a way that the number of arithmetic operations are essentially the same as the point SOR method. For these reasons, point SOR is rarely, if ever, used in commercial simulators. In fact, of the iterative techniques discussed in this chapter, SIP and block SOR are most frequently used.