As before, $\underline{G}_k$ is tridiagonal and $\underline{C}_k$ and $\underline{D}_k$ are diagonal. However, the cancellation that led to (6.41) does not occur with (6.48).

In applying the Fourier series representation to (6.47) a dense system of equations for $U_{s,k}$ replaces the tridiagonal system (6.44). Consequently the Fourier series representation applied to (6.47) is no more economical than Gauss elimination (Sect. 6.2.1) applied directly to (6.47). Depending on the grid, (6.47) may also include non-zero contributions associated with grid points $(j-1, k+1)$, $(j-1, k-1)$, $(j+1, k+1)$ and $(j+1, k-1)$.

6.3 Iterative Methods

Iterative methods can be applied directly to the nonlinear system of equations (6.1); however, it is easier to generalise iterative techniques by considering the linear form (6.2). All iterative techniques can be viewed as procedures for successively modifying an initial guess so that the solution is systematically approached. Generally, simple iterative methods will not converge unless $\underline{A}$, in (6.2), has large elements on the main diagonal (6.52).

6.3.1 General Structure

The general structure of stationary iterative techniques is illustrated by rewriting (6.2) as

$$(\underline{N}-\underline{P})V=B\ , \tag{6.49}$$

where $\underline{N}$ is close to $\underline{A}$ in some sense, i.e. $\|\underline{N}\|\approx\|\underline{A}\|$, but computationally efficient to factorise, e.g. $\underline{N}$ might be tridiagonal. Equation (6.49) can be rewritten as

$$\underline{N}V^{(n+1)}=\underline{P}V^{(n)}+B \quad \text{or}$$

$$V^{(n+1)}=\underline{N}^{-1}\underline{P}V^{(n)}+\underline{N}^{-1}B \quad \text{or} \tag{6.50}$$

$$V^{(n+1)}=V^{(n)}-\underline{N}^{-1}R^{(n)}\ , \tag{6.51}$$

where $R^{(n)}$ is the vector of equation residuals at the nth step of the iteration, i.e.

$$R^{(n)}=\underline{A}V^{(n)}-B\ .$$

As the exact solution is approached $\|R^{(n)}\|$ tends to zero, so that monitoring $\|R^{(n)}\|$ indicates the progress towards convergence. The general iterative method consists of an initial guess, $V^{(1)}$, and the successive improvement using (6.51). The various methods differ, primarily, in the choice for $\underline{N}$.

The scheme (6.51) will converge if the spectral radius, (i.e. the magnitude of the maximum eigenvalue) of $\underline{N}^{-1}\underline{P}$ is less than unity. This condition often corresponds to the more restrictive condition that $\underline{A}$ is diagonally dominant, i.e.

$$|A_{jj}|>\sum_{\substack{i\\ i\neq j}}|A_{ij}|\ . \tag{6.52}$$

The Jacobi method is one of the simpler iterative schemes for solving (6.2). In this method

$$\underline{N}=\underline{DI} \quad \text{and} \quad \underline{P}=(\underline{L}+\underline{U})\ ,$$

where $\underline{DI}$ is the diagonal matrix and $\underline{L}$ and $\underline{U}$ are the strictly lower and upper triangular matrices respectively. Thus (6.50) becomes

$$v_i^{(n+1)}=\left(B_i-\sum_{\substack{j\\ j\neq i}}^{N}A_{ij}v_j^{(n)}\right)\Big/A_{ii}\ . \tag{6.53}$$

The Jacobi method in the above form is uneconomic, requiring too many iterations to reach convergence. However, it can be significantly improved by either Chebyshev acceleration (Hageman and Young 1981, Chap. 4) or conjugate gradient acceleration (Sect. 6.3.4).

A more immediate improvement to the Jacobi method is provided by the Gauss-Seidel method in which values of $v_j^{(n+1)}$ are used on the right-hand side of (6.53) as soon as they are available.

In this case

$$\underline{N}=\underline{DI}-\underline{L}\ , \quad \underline{P}=\underline{U} \tag{6.54}$$

and the equivalent of (6.53) is

$$v_i^{(n+1)}=\left(B_i-\sum_{j=1}^{i-1}A_{ij}v_j^{(n+1)}-\sum_{j=i+1}^{N}A_{ij}v_j^{(n)}\right)\Big/A_{ii}\ . \tag{6.55}$$

Gauss-Seidel iteration is typically twice as fast as Jacobi iteration but cannot be accelerated. If $\underline{A}$ is diagonally dominant, (6.52), the convergence of Jacobi and Gauss-Seidel methods is guaranteed.

Successive overrelaxation (SOR) provides a significant improvement over Gauss-Seidel by evaluating $v^{(n+1)}$ as a weighted average of $v_i^{(n)}$ and $(v_i^{(n+1)})_{GS}$. Thus the SOR scheme can be written as

$$v_i^{(n+1)}=\lambda\left[\left(B_i-\sum_{j=1}^{i-1}A_{ij}v_j^{(n+1)}-\sum_{j=i+1}^{N}A_{ij}v_j^{(n)}\right)\Big/A_{ii}\right]+(1-\lambda)v_i^{(n)}\ . \tag{6.56}$$

The term λ is the relaxation parameter. The restriction $0<\lambda<2$, is necessary for convergence of SOR. In terms of (6.51) the SOR scheme corresponds to

$$\underline{N}=\frac{\underline{DI}}{\lambda}-\underline{L}\ . \tag{6.57}$$

The number of iterations for convergence is sensitive to the choice of λ. An optimum choice is given by

$$\lambda_{\text{opt}}=\frac{2}{1+(1-\mu^2)^{1/2}}\ , \tag{6.58}$$

where μ is the largest eigenvalue of $\underline{I}-\underline{D}\underline{I}^{-1}\underline{A}$. However, finding μ explicitly can be as expensive as solving the original problem. Therefore a preferred strategy is to obtain an estimate of μ as the iteration (6.56) proceeds. Equation (6.58) then provides an improved value for λ. Details are provided by Hageman and Young (1981, Chap. 9).

The SOR scheme with a good estimate of λ_{opt} is considerably more efficient than either the Jacobi or Gauss-Seidel methods. However, in the original form it is not possible to apply acceleration techniques, Chebyshev or conjugate gradient, to SOR. But by making a small modification to give the symmetric successive overrelaxation (SSOR) method it is possible to introduce acceleration techniques. The SSOR method consists of two stages. In the first the SOR scheme is used. In the second stage the unknowns are iterated in the reverse order using the SOR scheme with the same value of λ as in the first stage. The SSOR method corresponds to the following choice for $\underline{N}$ in (6.51):

$$\underline{N}=\frac{(\underline{D}\underline{I}+\lambda\underline{L})\underline{D}\underline{I}^{-1}(\underline{D}\underline{I}+\underline{U})}{\lambda(2-\lambda)}\ . \tag{6.59}$$

It should be emphasised that the SSOR method is less efficient than the SOR method, unless acceleration techniques are included.

6.3.2 Duct Flow by Iterative Methods

In this section we apply the three iterative methods, Jacobi, Gauss-Seidel and successive over-relaxation (SOR), to the problem of fully developed laminar flow in a square duct. These three iterative methods are discussed in Sect. 6.3.1, as explicit or point algorithms. Using the Thomas algorithm (subroutines BANFAC and BANSOL, Sect. 6.2.3), these iterative methods can also be applied as implicit or line algorithms; line SOR and ADI will also be described in this section.

The current example, the duct-flow problem, is also used to illustrate the finite element method in Sect. 5.5.2 and a computer program, DUCT (Fig. 5.22), is provided. Viscous flow in a duct is governed by the nondimensional equation (5.97)

$$\left(\frac{b}{a}\right)^2\frac{\partial^2\bar{w}}{\partial x^2}+\frac{\partial^2\bar{w}}{\partial y^2}+1=0\ , \tag{6.60}$$

with boundary conditions $\bar{w}=0$ at $x=\pm1$, $y=\pm1$. The parameter b/a is the duct aspect ratio, Fig. 5.21. A three-point finite difference discretisation of (6.60) is

$$\left(\frac{b}{a}\right)^2\frac{w_{j-1,k}-2w_{j,k}+w_{j+1,k}}{\Delta x^2}+\frac{w_{j,k-1}-2w_{j,k}+w_{j,k+1}}{\Delta y^2}+1=0\ . \tag{6.61}$$

Applying the Jacobi scheme (Sect. 6.3.1) to (6.61) produces the algorithm

$$w_{j,k}^{(n+1)}=0.5\left[1+\left(\frac{b}{a}\right)^2\frac{w_{j-1,k}^{(n)}+w_{j+1,k}^{(n)}}{\Delta x^2}+\frac{w_{j,k-1}^{(n)}+w_{j,k+1}^{(n)}}{\Delta y^2}\right]\Big/\text{PAR1}\ , \tag{6.62}$$

where (n) is the iteration index and $\text{PAR1}=(b/a)^2/\Delta x^2+1/\Delta y^2$. In the Jacobi scheme all grid-point values on the right-hand side of (6.62) are evaluated at the nth iteration level.

In the Gauss-Seidel scheme grid-point values are evaluated at the latest iteration level available. Thus if the iteration sweeps repeatedly in the y direction (increasing k) for successive x values (increasing j) the Gauss-Seidel equivalent of (6.62) would be

$$w_{j,k}^{(n+1)}=0.5\left[1+\left(\frac{b}{a}\right)^2\frac{w_{j-1,k}^{(n+1)}+w_{j+1,k}^{(n)}}{\Delta x^2}+\frac{w_{j,k-1}^{(n+1)}+w_{j,k+1}^{(n)}}{\Delta y^2}\right]\Big/\text{PAR1}\ . \tag{6.63}$$

In the SOR scheme the solution of (6.63), now called $w_{j,k}^{(*)}$, is combined with the previous solution, $w_{j,k}^{(n)}$, as

$$w_{j,k}^{(n+1)}=\lambda w_{j,k}^{(*)}+(1-\lambda)w_{j,k}^{(n)}=w_{j,k}^{(n)}+\lambda(w_{j,k}^{(*)}-w_{j,k}^{(n)})\ , \tag{6.64}$$

where λ is the relaxation factor. The Gauss-Seidel scheme is obtained when $\lambda=1$. The SOR scheme applied to (6.61) in the range $0<\lambda<2$ will produce a converged solution.

The three iterative methods are also applied with the finite element discretisation of (6.60), which is described in Sect. 5.5.2. For example, the equivalent of (6.63) is given by (5.108).

Table 6.3 provides a comparison of the number of iterations to reach convergence for the duct-flow problem, solved by the program DUCT, on an 11×11 grid.

Table 6.3. Number of iterations to convergence (b/a = 1.0)

λ	Point FDM (TOL = 1×10^{-5})	Point FDM (TOL = 1×10^{-6})	Point FEM (TOL = 1×10^{-6})	ADI-FDM (TOL = 1×10^{-6})	ADI-FEM (TOL = 1×10^{-6})	λ_y
Jacobi	41	87	64	19	19	0.9
0.8	39	74	55	16	18	1.1
0.9	33	62	45	14	17	1.3
1.0 (GS)	28	51	38	12	16	1.7
1.1	24	43	32	12	15	2.1
1.2	21	36	26	12	14	2.5
1.3	18	29	21	12	13	2.8
1.4	15	23	17	11	14	3.3
1.5	12	18	12	11	15	3.7
1.55	11	15	13	11	16	4.1
1.6	11	17	15	12	17	4.5
1.7	14	21	20			
1.8	21	28	29			

Convergence is assumed when $\|R^{(n)}\|_{rms} < TOL$. Results for two tolerances are presented. The starting solution for the iteration is the exact solution of (6.60). For a finite grid this solution is not the same as the solution of the discretised equations (6.61). Starting from the solution $w_{j,k}^{(0)}=0$ requires more iterations, as can be appreciated by comparing the results shown in Tables 6.3 and 6.4.

The results shown in Table 6.3 indicate that the SOR scheme, with the optimum value of λ, converges more quickly than the Gauss-Seidel scheme, which converges more quickly than the Jacobi scheme. For the finite difference method the reduction of the rms value of the change in the solution, $w_{j,k}^{(n+1)}-w_{j,k}^{(n)}$, is plotted against iteration number n in Fig. 6.20.

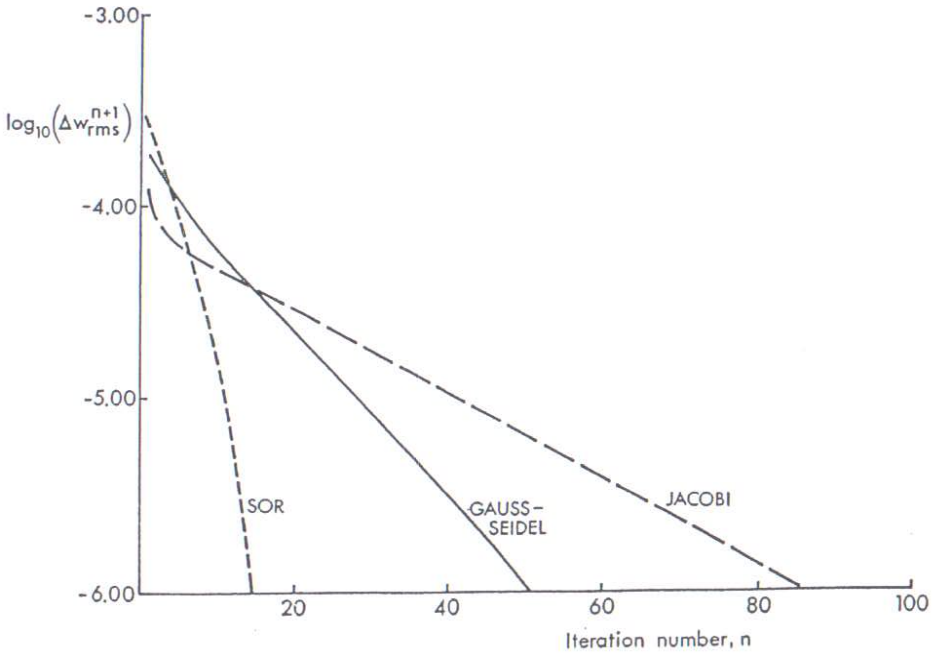

Fig. 6.20. Convergence history for point FDM (TOL $=1\times10^{-6}$)

The results shown in Table 6.3 indicate that the point finite element method (5.108, 109) converges more rapidly than the point finite difference method for values of $\lambda \leqq 1.7$. However the computational work per iteration is greater for the finite element method, so that it is not necessarily more efficient than the finite difference method for this problem.

All the algorithms given above, that is (6.62–64) have been explicit. It is possible, and desirable, to consider groups of nodal unknowns implicitly, as long as the resultant system of equations can be solved efficiently. This can be done by forming tridiagonal systems of equations associated with each line (constant k) in the x direction. Thus (6.61) is then written as an algorithm as

$$\left(-\frac{(b/a)^2}{\Delta x^2}\right)w_{j-1,k}^{(*)}+\left(\frac{2(b/a)^2}{\Delta x^2}\right)w_{j,k}^{(*)}-\left(\frac{(b/a)^2}{\Delta x^2}\right)w_{j+1,k}^{(*)}$$
$$=1+\frac{w_{j,k-1}^{(n)}-2w_{j,k}^{(n)}+w_{j,k+1}^{(n)}}{\Delta y^2}\ . \qquad (6.65)$$

This system can be solved conveniently using BANFAC/BANSOL (Sect. 6.2.3). Since the coefficients multiplying $w^{(*)}$ are constant on the left-hand side of (6.65), BANFAC is only required for the grid-line $k=2$. The same factorisation is used by BANSOL for all grid-lines. The combination of (6.65) and (6.64) is known as successive line overrelaxation (SLOR).

The ADI formulation (Varga 1962, Chap. 7) is similar to SLOR but with the implicit direction alternating at each iteration. Thus in place of (6.65) the following two-stage algorithm is used. For the first stage the following tridiagonal system associated with each line (constant k) in the x direction is solved:

$$-\lambda_x w_{j-1,k}^{(*)}+(1+2\lambda_x)w_{j,k}^{(*)}-\lambda_x w_{j+1,k}^{(*)}=\lambda_y\Delta y^2+\lambda_y w_{j,k-1}^{(n)}$$
$$+(1-2\lambda_y)w_{j,k}^{(n)}+\lambda_y w_{j,k+1}^{(n)}\ . \qquad (6.66)$$

For the second stage the following tridiagonal system associated with each line (constant j) in the y direction is solved:

$$-\lambda_y w_{j,k-1}^{(n+1)}+(1+2\lambda_y)w_{j,k}^{(n+1)}-\lambda_y w_{j,k+1}^{(n+1)}=\lambda_y\Delta y^2+\lambda_x w_{j-1,k}^{(*)}$$
$$+(1-2\lambda_x)w_{j,k}^{(*)}+\lambda_x w_{j+1,k}^{(*)}\ , \qquad (6.67)$$

where λ_y is chosen to accelerate convergence and $\lambda_x=(b\Delta y/a\Delta x)^2\lambda_y$. Strategies for choosing λ_y are discussed by Varga (1962) and Wachpress (1966).

The above ADI scheme is equivalent to the ADI scheme described in Sect. 8.2.1. Comparable formulae to (6.66, 67) can be obtained for the finite element method by adapting (8.35, 36). The number of iterations for the ADI finite difference method and ADI finite element method are shown in Table 6.3. Both methods give better convergence than the corresponding point algorithms for λ not at the optimum value.

The convergence rate is a strong function of grid refinement. As the grid is refined the number of iterations to convergence increases and the optimum λ becomes larger. This situation is illustrated in Table 6.4, which is based on a starting solution $w_{j,k}^{(0)}=0$.

The iterative algorithms considered above can be written as

$$\boldsymbol{w}^{(n+1)}=\underline{\boldsymbol{G}}\,\boldsymbol{w}^{(n)}\ .$$

The reduction in the rate of convergence as the grid is refined corresponds to the maximum eigenvalue of $\underline{\boldsymbol{G}}$ approaching unity. Some means of accelerating the convergence rate is clearly desirable. Hageman and Young (1981) discuss conjugate gradient and Chebyshev acceleration techniques. However, applied to Jacobi or

Table 6.4. Effect of grid refinement on convergence: Duct problem by point SOR (FDM)

Grid	Optimum λ	Number of iterations to convergence (TOL $= 1 \times 10^{-6}$)
6×6	1.30	12
11×11	1.55	23
21×21	1.74	41
41×41	1.86	79

Gauss-Seidel schemes neither Chebyshev nor conjugate gradient acceleration produce a faster rate of convergence than SOR with an optimum value of the relaxation factor λ. As a very robust algorithm, Hageman and Young recommend SOR with an adaptive algorithm for choosing λ. A subroutine for this purpose is provided by Hageman and Young (1981, pp. 368–372). Jennings (1977a, pp. 210–212) discusses the use of Aitken acceleration.

It should be pointed out that for iterative techniques, such as those described above, to be effective the problem being solved should be strongly elliptic, e.g. second-order derivatives are present but not first-order derivatives. Often this leads to the matrix of equations, e.g. those formed from (6.61), being symmetric and positive definite. The matrix $\underline{A}$ is symmetric and positive definite if $\underline{A}^T = \underline{A}$ and $x^T \underline{A} x > 0$ for all $x \neq 0$.

Examples of equations that are effectively solved by the above iterative techniques are the Poisson equation (for the stream function or pressure) that arises in viscous flow problems based on the stream function vorticity formulation (Sect. 17.3) and the potential equation that often arises in inviscid flow problems (Chap. 14), particularly transonic flow (Sect. 14.3).

If first derivatives of significant magnitude are present the above techniques are very ineffective and may fail to converge. For example, applying the point SOR (FDM) technique to the two-dimensional Burgers' equations considered in Sect. 6.1.3, on a 21×11 grid, requires $\lambda \leqq 1.0$ for convergence. After 270 iterations the equation residuals are reduced to about 1.4×10^{-5}. For comparison, an approximate factorisation of the unsteady Newton's method (Sect. 6.4.1) achieves the same degree of convergence in 15 iterations.

Significant first derivatives occur in the momentum equations for the primitive variables (Sect. 17.1), the transport equation for the vorticity (Sect. 17.3) and the energy equation (Sect. 11.2.4).

6.3.3 Strongly Implicit Procedure

This strongly implicit procedure (SIP) scheme starts from the general stationary iterative technique (6.49) and factorises $\underline{N}$ into $\underline{L}\ \underline{U}$ for the situation where $\underline{A}$ arises from three-point centred difference discretisations in two dimensions. In this case $\underline{A}$ has the structure shown in Fig. 6.1.

Stone (1968) showed that this could be done in a way that produces three diagonals in $\underline{L}$ and three diagonals in $\underline{U}$ of which the main diagonal elements are unity. Forming $\underline{N} = \underline{L}\,\underline{U}$ indicates that $\underline{P}$ consists of two diagonals multiplying $v_{i+1,j-1}$ and $v_{i-1,j+1}$. For a linear system of equations (6.2) the elements of $\underline{L}$, $\underline{U}$ and $\underline{P}$ can be evaluated once and for all.

The algorithm is implemented in two stages. First a forward sweep solves

$$\underline{L} w^{(n+1)} = B + \underline{P} V^{(n)} \ . \tag{6.68}$$

This is followed by a back-substitution

$$\underline{U} V^{(n+1)} = w^{(n+1)} \ . \tag{6.69}$$

The algorithm is robust and is often more efficient than the ADI scheme (6.66, 67).

Schneider and Zedan (1981) have developed a modified strongly implicit (MSI) algorithm that is applicable to either five-point or nine-point two-dimensional discretisations, i.e. it can be applied to three-point finite difference discretisations or linear finite element discretisations. Here the five-point version of MSI will be described; the reader is referred to Schneider and Zedan for the nine-point version.

The MSI algorithm is executed using (6.68 and 69). However $\underline{L}$, $\underline{U}$ and $\underline{P}$ differ from the form used by Stone. In the MSI algorithm $\underline{L}$ has four non-zero entries per row, $\underline{U}$ has three non-zero off-diagonal entries plus unity on the diagonal and $\underline{P}$ has two entries. If V is numbered in the increasing j and then increasing k directions, (6.68) becomes

$$\begin{aligned} f_{j,k} w_{j,k}^{(n+1)} = b_{j,k} &+ p_{j,k}^1 [v_{j+2,k-1} - \alpha(-2v_{j,k} + 2v_{j+1,k} + v_{j,k-1})]^{(n)} \\ &+ p_{j,k}^2 [v_{j-2,k+1} - \alpha(-2v_{j,k} + 2v_{j-1,k} + v_{j,k+1})]^{(n)} \\ &- [c_{j,k} v_{j,k-1} + d_{j,k} v_{j+1,k-1} + e_{j,k} v_{j-1,k}]^{(n+1)} \ . \end{aligned} \tag{6.70}$$

The back-substitution (6.69) is implemented as

$$v_{j,k}^{(n+1)} = w_{j,k}^{(n+1)} - (g_{j,k} v_{j+1,k} + h_{j,k} v_{j-1,k+1} + t_{j,k} v_{j,k+1})^{(n+1)} \ . \tag{6.71}$$

In (6.70 and 71) coefficients c, d, e and f are part of $\underline{L}$, coefficients g, h and t are the part of $\underline{U}$ and coefficients p^1 and p^2 are part of $\underline{P}$. The coefficients are related to the elements of $\underline{A}$ by

$$\begin{aligned} &c_{j,k} = \frac{a_{j,k}}{1 - \alpha g_{j,k-1} g_{j+1,k-1}} \ , \quad d_{j,k} = -c_{j,k} g_{j,k-1} \ , \quad e_{j,k} = a_{j,k} - c_{j,k} h_{j,k-1} \ , \\ &p_{j,k}^1 = d_{j,k} g_{j+1,k-1} \ , \quad p_{j,k}^2 = e_{j,k} h_{j-1,k} \ , \\ &f_{j,k} = a_{j,k} - c_{j,k} t_{j,k-1} - d_{j,k} h_{j+1,k-1} - e_{j,k} g_{j-1,k} + 2\alpha(p_{j,k}^1 + p_{j,k}^2) \ , \\ &g_{j,k} = a_{j+1,k} - d_{j,k} t_{j+1,k-1} - 2\alpha p_{j,k}^1 \ , \quad h_{j,k} = \frac{-e_{j,k} t_{j-1,k}}{f_{j,k}} \ , \\ &t_{j,k} = \frac{a_{j,k+1} - \alpha p_{j,k}}{f_{j,k}} \ . \end{aligned} \tag{6.72}$$