

# Chapter 9

## Solution Techniques for Elliptic Problems

### 9.1 Direct Solution Methods

In Sections 8.1, 2, we saw that the discretization of an elliptic partial differential equation led to the solution of a large, sparse, linear algebraic system. In this chapter, we address the solution of such systems. A slight change in notation will simplify the presentation. Thus, our goal is to solve linear algebraic systems of the form

$$\mathbf{Ax} = \mathbf{b}, \tag{9.1.1}$$

where  $\mathbf{A}$  is a large, sparse, and typically positive-definite  $N \times N$  matrix. In this section, we study *direct* techniques where the solution of (9.1.1) is found after a finite number of algebraic operations. In subsequent sections, we'll consider *iterative* techniques where the solution of (9.1.1) will only be found as the number of steps becomes infinite.

Direct solution utilize Gaussian elimination and its many variants. Pivoting is unnecessary with positive-definite systems and we shall assume that this is the case here. Gaussian elimination is regarded as a factorization of  $\mathbf{A}$  into the product

$$\mathbf{A} = \mathbf{LU} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{N1} & l_{N2} & \cdots & \cdots & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1N} \\ & u_{22} & \cdots & u_{2N} \\ & & \ddots & \vdots \\ & & & u_{NN} \end{bmatrix}. \tag{9.1.2}$$

The matrix  $\mathbf{L}$  is lower triangular and  $\mathbf{U}$  is upper triangular. Zero elements above the

diagonal of  $\mathbf{L}$  and below the diagonal of  $\mathbf{U}$  are not shown. Expanding (9.1.2)

$$u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}, \quad j = i, i+1, \dots, N, \quad (9.1.3a)$$

$$l_{ji} = \frac{1}{u_{ii}}(a_{ji} - \sum_{k=1}^{i-1} l_{jk}u_{ki}), \quad j = i+1, i+2, \dots, N, \quad i = 1, 2, \dots, N. \quad (9.1.3b)$$

The summations involved in (9.1.3) are understood to be zero if the lower limit exceeds the upper one.

Once  $\mathbf{L}$  and  $\mathbf{U}$  have been determined, (9.1.1) may be solved by forward and backward substitution; thus, we have

$$\mathbf{Ax} = \mathbf{LUx} = \mathbf{b}.$$

Let

$$\mathbf{Ux} = \mathbf{y} \quad (9.1.4a)$$

then

$$\mathbf{Ly} = \mathbf{b}. \quad (9.1.4b)$$

Expressed in scalar form, the forward substitution (9.1.4b) and the backward substitution (9.1.4a) are

$$y_i = b_i - \sum_{k=1}^{i-1} l_{ik}y_k, \quad i = 1, 2, \dots, N, \quad (9.1.5a)$$

$$x_i = \frac{1}{u_{ii}}(y_i - \sum_{k=i+1}^N u_{ik}x_k), \quad i = N, N-1, \dots, 1. \quad (9.1.5b)$$

The procedure fails if  $u_{ii} = 0$ ,  $i = 1, 2, \dots, N$ . Pivoting may be necessary if this should occur.

The above procedures ignore sparsity in  $\mathbf{A}$  which, as we'll show, is not practical. As a first step towards this end, let us consider the banded structure of  $\mathbf{A}$ .

**Definition 9.1.1.** A matrix  $\mathbf{A}$  is called a band matrix of bandwidth  $p + q + 1$  if  $a_{ij} = 0$  for  $j > i + p$  and  $i > j + q$ .

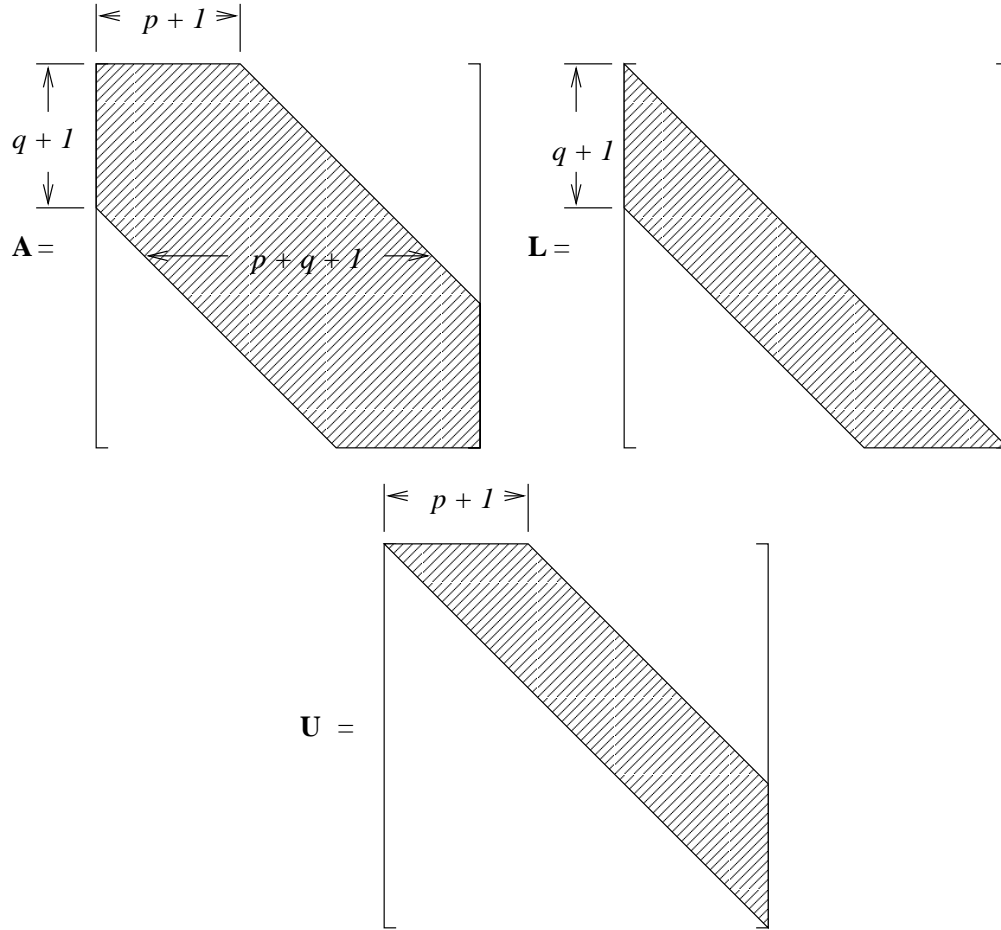


Figure 9.1.1: Structure of a band matrix  $\mathbf{A}$  of bandwidth  $p + q + 1$  (left) and of its lower and upper triangular factors  $\mathbf{L}$  (right) and  $\mathbf{U}$  (bottom), respectively. Elements not in the shaded regions are zero.

When using Gaussian elimination with a band matrix, it is easily shown that the factors  $\mathbf{L}$  and  $\mathbf{U}$ , respectively, have the structures of the lower and upper portions of  $\mathbf{A}$  (Figure 9.1.1); thus,  $l_{ij} = 0$  for  $i > j + q$  and  $j > i$  and  $u_{ij} = 0$  for  $i > j$  and  $j > i + p$ .

For a band matrix with  $q = p$ , the factorization (9.1.3) and forward and backward substitution (9.1.5) phases of Gaussian elimination become

$$u_{ij} = a_{ij} - \sum_{k=\max(1, j-p)}^{i-1} l_{ik} u_{kj}, \quad j = i, i + 1, \dots, i + p, \quad (9.1.6a)$$

$$l_{ji} = \frac{1}{u_{ii}} \left( a_{ji} - \sum_{k=\max(1, j-p)}^{i-1} l_{jk} u_{ki} \right), \quad j = i + 1, i + 2, \dots, i + p, \quad i = 1, 2, \dots, N. \quad (9.1.6b)$$

$$y_i = b_i - \sum_{k=\max(1,i-p)}^{i-1} l_{ik} y_k, \quad i = 1, 2, \dots, N, \quad (9.1.7a)$$

$$x_i = \frac{1}{u_{ii}} \left( y_i - \sum_{k=i+1}^{\min(N,i+p)} u_{ik} y_k \right), \quad i = N, N-1, \dots, 1. \quad (9.1.7b)$$

The algorithm (9.1.6,9.1.7) ignores embedded zeros within the band. Accounting for these is typically not necessary with a direct solution method since, as will be discussed, they become nonzero during the elimination process.

The reduction in time and space complexity is significant when  $p \ll N$ . Approximate operation counts for the factorization and forward and backward substitution phases of the full and banded procedures are given in Table 9.1.1. In order to provide some meaning to these estimates, consider the solution of a Dirichlet problem for Laplace's equation on a square. With  $\Delta x = \Delta y$  the resulting algebraic system (8.1.4) has the form shown in Figure 9.1.2. Using (8.1.2), reveals that the diagonal elements of  $\mathbf{A}$  are unity and all off-diagonal terms are  $-1/4$ . Each block of  $\mathbf{A}$  is  $(J-1) \times (J-1)$  and there are  $J-1$  blocks. Thus,  $N = (J-1)^2$  and  $p = J-1$ . Using this data with the estimates shown in Table 9.1.1 gives the approximate operation counts reported on the right of Table 9.1.1. Even modest values of  $J$  indicate the impracticality of ignoring the sparsity present in  $\mathbf{A}$ .

	Full	Banded		Full	Banded
Factor	$N^3/3$	$Np(p+1)$	Factor	$J^6/3$	$J^4$
Solve	$N^2$	$N(2p+1)$	Solve	$J^4$	$2J^3$

Table 9.1.1: Approximate operation counts for solving full and banded  $N \times N$  linear systems having a bandwidth of  $2p+1$  by Gaussian elimination (left). Approximate operation counts when solving a Dirichlet problem for Laplace's equation on a  $J \times J$  square mesh (right) which has  $N = (J-1)^2$  and  $p = J-1$ .

The factorization of  $\mathbf{A}$  creates nonzero entries within the band. Hence, for the Laplacian operator, the storage needed for  $\mathbf{L}$  and  $\mathbf{U}$  using banded Gaussian elimination is approximately  $2J^3$ , while the nonzero entries of  $\mathbf{A}$  only require  $5J^2$  memory locations.

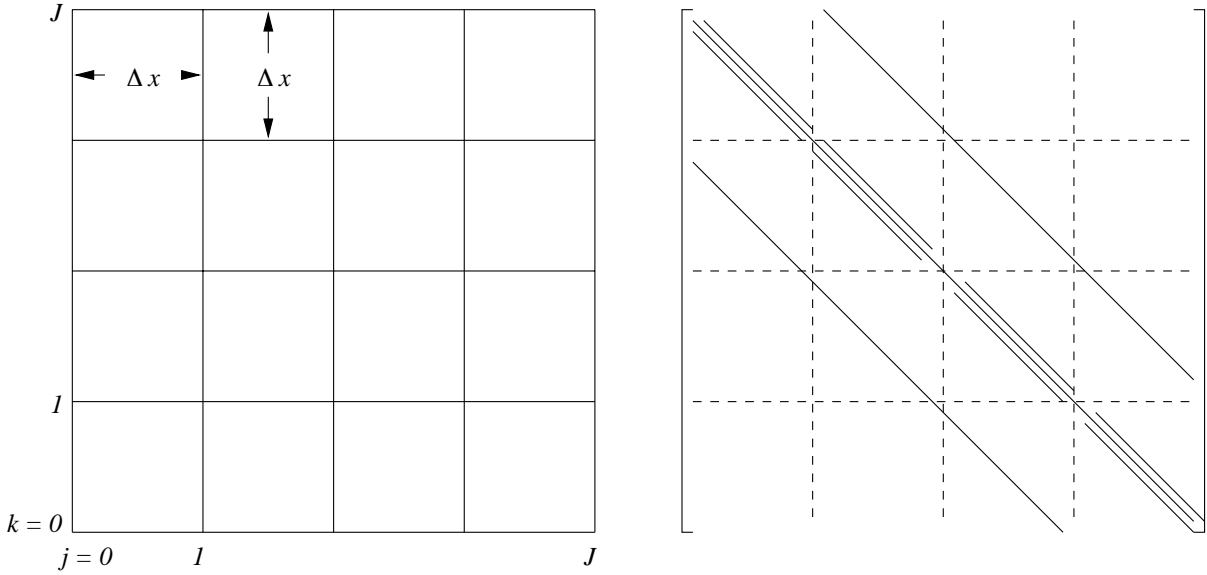


Figure 9.1.2: Uniform square mesh (left) and structure of the corresponding matrix  $\mathbf{A}$  for the solution of Laplace's equation using centered finite differences (8.1.2).

We will have to use iterative methods in order to take full advantage of the sparsity in  $\mathbf{A}$ . Nevertheless, some additional time and space savings are possible. For example, a symmetric, positive definite matrix  $\mathbf{A}$  may be factored as

$$\mathbf{A} = \mathbf{L}\mathbf{D}\mathbf{L}^T \tag{9.1.8a}$$

where

$$\mathbf{L} = \begin{bmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & & & \\ \vdots & \vdots & \ddots & & \\ l_{N1} & l_{N2} & \cdots & & 1 \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} d_1 & & & & \\ & d_2 & & & \\ & & \ddots & & \\ & & & & d_N \end{bmatrix}. \tag{9.1.8b}$$

Computing the product in (9.1.8a) using (9.1.8b) yields

$$d_i = a_{ii} - \sum_{k=1}^{i-1} d_k l_{ik}^2, \tag{9.1.9a}$$

$$l_{ji} = \frac{1}{d_i} (a_{ji} - \sum_{k=1}^{i-1} d_k l_{jk} l_{ik}), \quad j = i + 1, i + 2, \dots, N, \quad i = 1, 2, \dots, N. \tag{9.1.9b}$$

The solution phase follows by substituting (9.1.8a) into (9.1.1) to get

$$\mathbf{A}\mathbf{x} = \mathbf{L}\mathbf{D}\mathbf{L}^T\mathbf{x} = \mathbf{b}.$$

Letting

$$\mathbf{L}^T \mathbf{x} = \mathbf{y}, \quad \mathbf{D} \mathbf{y} = \mathbf{z}, \quad \mathbf{L} \mathbf{z} = \mathbf{b}. \quad (9.1.10)$$

The solution is obtained after forward, diagonal, and backward substitution steps, which have the scalar form

$$z_i = b_i - \sum_{k=1}^{i-1} l_{ik} z_k, \quad i = 1, 2, \dots, N, \quad (9.1.11a)$$

$$y_i = z_i / d_i, \quad i = 1, 2, \dots, N, \quad (9.1.11b)$$

$$x_i = y_i - \sum_{k=i+1}^N l_{ki} x_k, \quad i = N, N-1, \dots, 1. \quad (9.1.11c)$$

Banded versions of the factorization and solution steps can also be developed.

The block tridiagonal algorithm [3] exploits the fact that  $\mathbf{A}$  has a tridiagonal structure with entries that are matrices (*cf.* (8.1.4)). This too has (approximately) the same number of operations as banded Gaussian elimination (9.1.6, 9.1.7). A different ordering of the equations and unknowns, however, can significantly reduce fill-in of the band and, hence, the order of operations. Nested dissection, developed by George [1, 2], is known to be optimal in certain situations. The dissection process is illustrated for a  $4 \times 4$  mesh in Figure 9.1.3. Alternate unknowns are eliminated first to create the coarser mesh of “macro elements” shown at the right of Figure 9.1.3. Midside nodes of these macro elements are eliminated next to leave, in this case, a single unknown at the center of the domain (bottom). Although we will not describe how to do the dissection for a more general mesh and problem, one can visualize the process and essential idea.

The structure of the linear systems obtained by using the row-by-row and nested-dissection orderings are shown in Figures 9.1.4 and 9.1.5, respectively, for the  $4 \times 4$  mesh of Figure 9.1.3. The matrix  $\mathbf{A}$  obtained by the nested-dissection ordering has a larger bandwidth than the one with the row-by-row ordering. However, even with this simple  $4 \times 4$  problem, we can see that the fill in is less with the nested dissection ordering than with the row-by-row ordering (Figures 9.1.4 and 9.1.5).

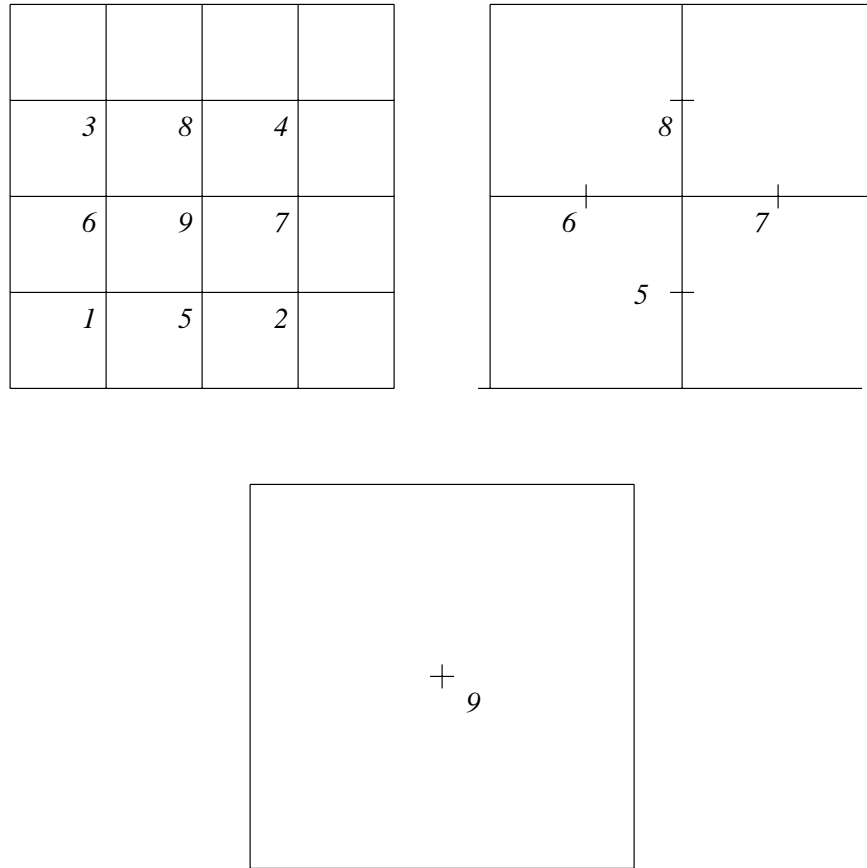


Figure 9.1.3: Nested dissection of a uniform  $4 \times 4$  mesh. Unknowns at the finest level (left) are eliminated first to create the coarser mesh (center). These unknowns are eliminated next to leave a single unknown (bottom).

A banded structure is not necessary for the efficient implementation of a Gaussian elimination procedure. It is just about as simple to implement a “skyline” or “profile” elimination strategy where the local bandwidth is used. This requires an additional vector indicating, *e.g.*, the number of leading zeros in a row or column. Let

$$l_{ij} = 0, \quad 0 \leq j < m_i^* \quad (9.1.12a)$$

and

$$k^* = \max(m_i^*, m_j^*) \quad (9.1.12b)$$

then the skyline form of the symmetric Cholesky decomposition (9.1.9) is

$$d_i = a_{ii} - \sum_{k=k^*}^{i-1} d_k l_{ik}^2, \quad (9.1.13a)$$

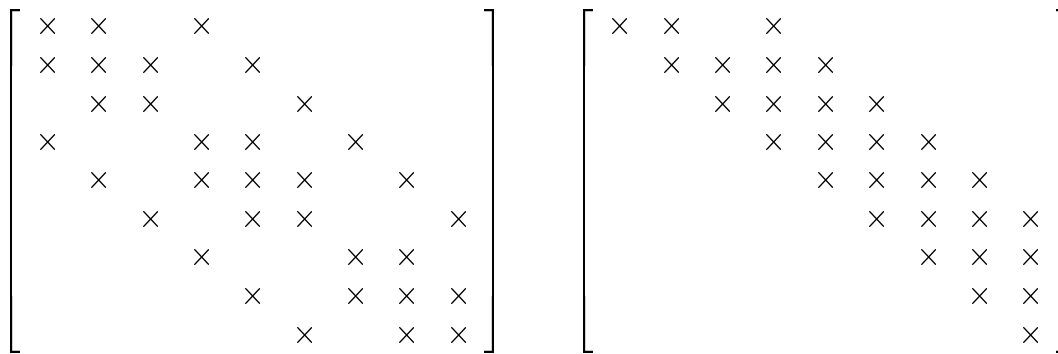


Figure 9.1.4: Matrix **A** when the finite difference equations for a  $4 \times 4$  mesh are ordered by rows (left) and the resulting fill-in of **U** (right).

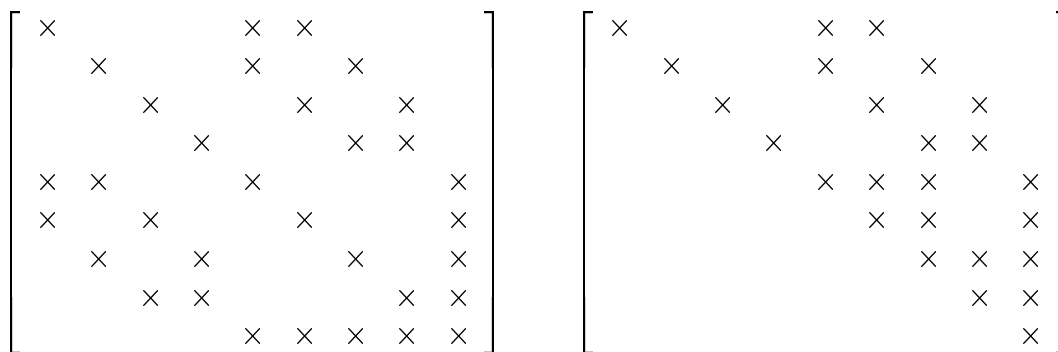


Figure 9.1.5: Matrix **A** when the finite difference equations have a nested dissection ordering (left) and the resulting fill-in of **U** (right).

$$l_{ji} = \frac{1}{d_i} \left( a_{ji} - \sum_{k=k^*}^{i-1} d_k l_{jk} l_{ik} \right), \quad j = m_i^* + 1, i + 2, \dots, N, \quad i = 1, 2, \dots, N. \quad (9.1.13b)$$

This procedure ignores any zeros within the profile of **L**. George [1] proved that the profile algorithm (9.1.12, 9.1.13) with the nested dissection ordering could solve a Dirichlet problem for Poisson’s equation on a  $J \times J$  square mesh in  $O(J^3)$  operations with  $O(J^2 \log_2 J)$  storage. These should be compared to the  $O(J^4)$  operations and  $O(J^3)$  storage required for the banded algorithm of Table 9.1.1. George [1] additionally showed that the nested ordering is optimal in the sense that all orderings of the mesh must yield an operation count of at least  $O(J^3)$ .

## 9.2 Basic Iterative Solution Methods

The direct methods of Section 9.1 require storage within the band or profile which could be significant for very large problems (*e.g.*, in excess of 10,000 equations). Storage and, perhaps, computer time can be reduced through the use of iterative techniques. As in Section 9.1, we'll focus on techniques for  $N \times N$  linear system having the form (9.1.1). Using a “fixed-point” strategy, we rewrite (9.1.1) in the form

$$\mathbf{x} = \mathbf{M}\mathbf{x} + \hat{\mathbf{b}} \quad (9.2.1)$$

and consider the iteration

$$\mathbf{x}^{(\nu+1)} = \mathbf{M}\mathbf{x}^{(\nu)} + \hat{\mathbf{b}}, \quad \nu = 0, 1, \dots \quad (9.2.2)$$

The iteration must be designed so that

$$\lim_{\nu \rightarrow \infty} \mathbf{x}^{(\nu)} = \mathbf{x}.$$

*Example 9.2.1.* Write

$$\mathbf{A} = \mathbf{A} + \mathbf{I} - \mathbf{I},$$

where  $\mathbf{I}$  is the  $N \times N$  identity matrix, and rewrite (9.1.1) as

$$\mathbf{I}\mathbf{x} = (\mathbf{I} - \mathbf{A})\mathbf{x} + \mathbf{b}.$$

This system has the form of (9.2.1) with

$$\mathbf{M} = \mathbf{I} - \mathbf{A}, \quad \hat{\mathbf{b}} = \mathbf{b}.$$

Here are two fundamental convergence criteria.

**Theorem 9.2.1.** *The iteration (9.2.2) converges to a fixed point  $\mathbf{x}$  of (9.2.1) when*

$$\|\mathbf{M}\| < 1. \quad (9.2.3)$$

*Proof.* Define

$$\mathbf{e}^{(\nu)} = \mathbf{x}^{(\nu)} - \mathbf{x} \quad (9.2.4)$$

and subtract (9.2.1) from (9.2.2) to obtain

$$\mathbf{e}^{(\nu+1)} = \mathbf{M}\mathbf{e}^{(\nu)}.$$

Thus,

$$\mathbf{e}^{(\nu+1)} = \mathbf{M}\mathbf{e}^{(\nu)} = \mathbf{M}^2\mathbf{e}^{(\nu-1)} = \dots = \mathbf{M}^{\nu+1}\mathbf{e}^{(0)}$$

or

$$\mathbf{e}^{(\nu)} = \mathbf{M}^{\nu}\mathbf{e}^{(0)}.$$

Taking a norm

$$\|\mathbf{e}^{(\nu)}\| = \|\mathbf{M}^{\nu}\mathbf{e}^{(0)}\| \leq \|\mathbf{M}^{\nu}\| \|\mathbf{e}^{(0)}\| \leq \|\mathbf{M}\|^{\nu} \|\mathbf{e}^{(0)}\|. \quad (9.2.5)$$

When  $\|\mathbf{M}\| < 1$ , we see that  $\|\mathbf{e}^{(\nu)}\| \rightarrow 0$  as  $\nu \rightarrow \infty$  and, hence, the iteration converges.  $\square$

**Theorem 9.2.2.** *The iteration (9.2.2) converges from any initial guess if and only if the spectral radius*

$$\rho(\mathbf{M}) \equiv \max_{1 \leq i \leq N} |\lambda_i(\mathbf{M})| < 1 \quad (9.2.6)$$

where  $\lambda_i$ ,  $i = 1, 2, \dots, N$ , are eigenvalues of the  $N \times N$  matrix  $\mathbf{M}$ .

*Proof.* From Lemma 3.3.1 we know

$$\rho^{\nu}(\mathbf{M}) = \rho(\mathbf{M}^{\nu}) \leq \|\mathbf{M}^{\nu}\|.$$

If the iteration (9.2.2) converges from any initial guess  $\mathbf{e}^{(0)}$ , then the results of Theorem 9.2.1 imply  $\|\mathbf{M}^{\nu}\| \rightarrow 0$  as  $\nu \rightarrow \infty$ ; hence,  $\rho(\mathbf{M}) < 1$ .

Proving that (9.2.2) converges when  $\rho(\mathbf{M}) < 1$  is slightly more involved. We'll establish the result when  $\mathbf{M}$  is diagonalizable. Isaacson and Keller [3], Section 1.1, establish the result under more general conditions.

If  $\mathbf{M}$  is diagonalizable, then there is a matrix  $\mathbf{P}$  such that

$$\mathbf{P}\mathbf{M}\mathbf{P}^{-1} = \mathbf{\Lambda}$$

where  $\mathbf{\Lambda}$  is a diagonal matrix. Now,

$$\mathbf{M}^{\nu} = (\mathbf{P}^{-1}\mathbf{\Lambda}\mathbf{P})(\mathbf{P}^{-1}\mathbf{\Lambda}\mathbf{P}) \dots (\mathbf{P}^{-1}\mathbf{\Lambda}\mathbf{P}) = \mathbf{P}^{-1}\mathbf{\Lambda}^{\nu}\mathbf{P}.$$

If  $|\lambda_i| < 1$ ,  $i = 1, 2, \dots, N$ , then

$$\lim_{\nu \rightarrow \infty} \|\mathbf{P}^{-1} \mathbf{A}^\nu \mathbf{P}\| = 0$$

and the iteration (9.2.2) converges.  $\square$

Theorems 9.2.1 and 9.2.2 prescribe convergence conditions. We also want an indication of the convergence rate of the iteration. Many measures are possible and we'll settle on the following.

**Definition 9.2.1.** *The average convergence rate of the iteration (9.2.2) is*

$$R_\nu(\mathbf{M}) \equiv -\frac{\ln \|\mathbf{M}^\nu\|}{\nu}. \quad (9.2.7a)$$

Using (9.2.5)

$$\|\mathbf{e}^{(\nu)}\| \leq \|\mathbf{M}^\nu\| \|\mathbf{e}^{(0)}\| = e^{-\nu R_\nu} \|\mathbf{e}^{(0)}\|.$$

Thus, convergence is fast when  $R_\nu$  is large or, equivalently, when  $\|\mathbf{M}\|$  is small. Additionally, since  $\rho(\mathbf{M}^\nu) \leq \|\mathbf{M}^\nu\|$  and  $\|\mathbf{M}^\nu\| \leq 1$  for a converging iteration,

$$R_\nu(\mathbf{M}) \leq -\ln \rho(\mathbf{M}). \quad (9.2.7b)$$

Thus, we may take  $-\ln \rho(\mathbf{M})$  as a measure of the convergence rate. Although the spectral radius is more difficult to compute than a matrix norm, this rate is independent of  $\nu$  and the particular norm.

Many iterative procedures partition  $\mathbf{A}$  as

$$\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{U} \quad (9.2.8a)$$

where

$$\mathbf{D} = \begin{bmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{NN} \end{bmatrix}, \quad \mathbf{L} = \begin{bmatrix} 0 & & & \\ -a_{21} & 0 & & \\ \vdots & \vdots & \ddots & \\ -a_{N1} & -a_{N2} & \cdots & 0 \end{bmatrix}, \quad (9.2.8b)$$

$$\mathbf{U} = \begin{bmatrix} 0 & -a_{12} & \cdots & -a_{1N} \\ & 0 & \cdots & -a_{2N} \\ & & \ddots & \vdots \\ & & & 0 \end{bmatrix}. \quad (9.2.8c)$$

### 9.2.1 Jacobi and Gauss-Seidel Iteration

Three classical iterative methods have the splitting defined by (9.2.8). With the *Jacobi method*, we solve for the diagonal terms of (9.1.1). Thus, using (9.2.8a), we write (9.1.1) in the form

$$(\mathbf{D} - \mathbf{L} - \mathbf{U})\mathbf{x} = \mathbf{b} \quad (9.2.9)$$

and consider the iteration

$$\mathbf{x}^{(\nu+1)} = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U})\mathbf{x}^{(\nu)} + \mathbf{D}^{-1}\mathbf{b} \quad (9.2.10a)$$

which has the form of (9.2.2) with

$$\mathbf{M}_J = \mathbf{D}^{-1}(\mathbf{L} + \mathbf{U}), \quad \hat{\mathbf{b}}_J = \mathbf{D}^{-1}\mathbf{b}. \quad (9.2.10b)$$

The scalar form (9.2.10) is

$$x_i^{(\nu+1)} = - \sum_{j=1, j \neq i}^N \frac{a_{ij}}{a_{ii}} x_j^{(\nu)} + \frac{b_i}{a_{ii}}, \quad i = 1, 2, \dots, N. \quad (9.2.11)$$

*Example 9.2.2.* The Jacobi method for the Poisson equation (8.1.2a) is

$$\begin{aligned} U_{jk}^{(\nu+1)} &= \theta_x(U_{j-1,k}^{(\nu)} + U_{j+1,k}^{(\nu)}) + \theta_y(U_{j,k-1}^{(\nu)} + U_{j,k+1}^{(\nu)}) + \theta_{xy}f_{jk}, \\ j &= 1, 2, \dots, J-1, \quad k = 1, 2, \dots, K-1. \end{aligned} \quad (9.2.12)$$

Updates to the solution at  $(j, k)$  are computed as a weighted average of solutions at its four neighboring points. Contrary to solutions obtained by direct methods, parallel computational techniques are easily used with Jacobi's method since the solution state at iteration  $\nu + 1$  is explicit.

It's easy to show that Jacobi iteration converges when  $\mathbf{A}$  satisfies some rather restrictive properties.

**Definition 9.2.2.** A matrix  $\mathbf{A}$  is strictly diagonally dominant if

$$\sum_{j=1, j \neq i}^N |a_{ij}| < |a_{ii}|, \quad i = 1, 2, \dots, N. \quad (9.2.13)$$

**Theorem 9.2.3.** *The Jacobi iteration converges in the maximum norm when  $\mathbf{A}$  is strictly diagonally dominant.*

*Proof.* If  $\mathbf{A}$  is strictly diagonally dominant, we may use (9.2.8) and (9.2.10) to obtain

$$\|\mathbf{M}_J\|_\infty = \max_{1 \leq i \leq N} \sum_{j=1, j \neq i}^N \frac{|a_{ij}|}{|a_{ii}|} < 1.$$

□

Convergence of Jacobi's method is too slow for practical serial computation, although it may be used for parallel computation. Gauss-Seidel iteration uses the latest solution information as soon as it becomes available. Thus, when computing  $x_i^{(\nu+1)}$  according to (9.2.11), we could use the latest iterates  $x_j^{(\nu+1)}$ ,  $j = 1, 2, \dots, i-1$ , on the right to obtain

$$x_i^{(\nu+1)} = - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(\nu+1)} - \sum_{j=i}^N \frac{a_{ij}}{a_{ii}} x_j^{(\nu)} + \frac{b_i}{a_{ii}}, \quad i = 1, 2, \dots, N. \quad (9.2.14)$$

In the matrix form of (9.2.9), this is equivalent to

$$(\mathbf{D} - \mathbf{L})\mathbf{x}^{(\nu+1)} = \mathbf{U}\mathbf{x}^{(\nu)} + \mathbf{b} \quad (9.2.15a)$$

which has the form of (9.2.2) with

$$\mathbf{M}_{GS} = (\mathbf{D} - \mathbf{L})^{-1}\mathbf{U}, \quad \hat{\mathbf{b}}_{GS} = (\mathbf{D} - \mathbf{L})^{-1}\mathbf{b}. \quad (9.2.15b)$$

*Example 9.2.3.* The Gauss-Seidel iteration for the Poisson equation (8.1.2a) is

$$\begin{aligned} U_{jk}^{(\nu+1)} &= \theta_x(U_{j-1,k}^{(\nu+1)} + U_{j+1,k}^{(\nu)}) + \theta_y(U_{j,k-1}^{(\nu+1)} + U_{j,k+1}^{(\nu)}) + \theta_{xy}f_{jk}, \\ &j = 1, 2, \dots, J-1, \quad k = 1, 2, \dots, K-1. \end{aligned} \quad (9.2.16)$$

The solution process depends on the order in which the equations are written. As described above and as shown in Figure 9.2.1, row ordering has been assumed.

*Example 9.2.4.* Consider the boundary value problem for Laplace's equation on a unit square  $\Omega$

$$\begin{aligned} \Delta u &= 0, \quad (x, y) \in \Omega, \\ u(x, y) &= \begin{cases} 0, & \text{if } x = 0, y = 1 \\ 1, & \text{if } x = 1, y = 0 \end{cases}, \quad (x, y) \in \partial\Omega. \end{aligned}$$

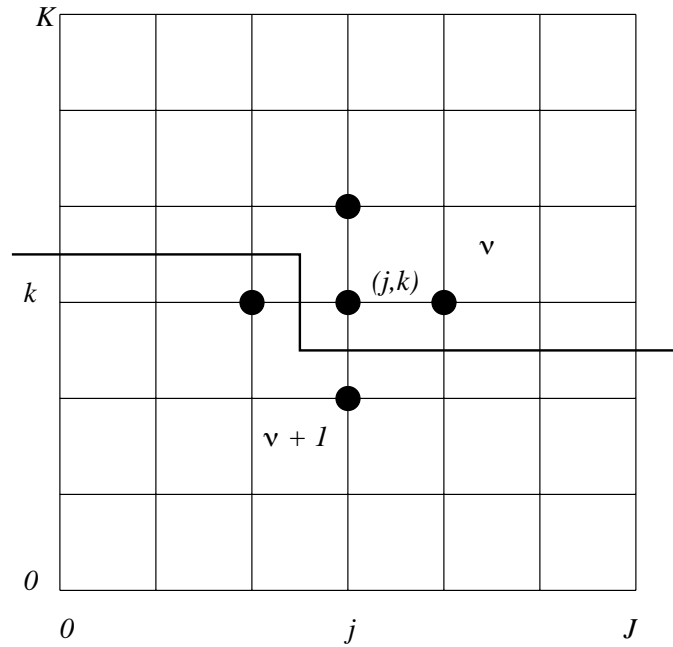


Figure 9.2.1: Gauss-Seidel iteration with row ordering.

Let us solve this problem on a  $3 \times 3$  mesh using Jacobi and Gauss-Seidel iteration with  $\Delta x = \Delta y = 1/3$ ; hence, using (8.1.2) with  $\theta_x = \theta_y = 1/4$  and  $f_{jk} = 0$ , we have

$$U_{jk} = \frac{1}{4}(U_{j+1,k} + U_{j-1,k} + U_{j,k+1} + U_{j,k-1}), \quad j, k = 1, 2.$$

The Jacobi iteration is

$$U_{jk}^{(\nu+1)} = \frac{1}{4}(U_{j+1,k}^{(\nu)} + U_{j-1,k}^{(\nu)} + U_{j,k+1}^{(\nu)} + U_{j,k-1}^{(\nu)}), \quad j, k = 1, 2, \quad \nu = 0, 1, \dots$$

Starting with the trivial initial guess  $U_{jk}^{(0)} = 0$ ,  $j, k = 1, 2$ , we present solutions after the one and five iterations in Table 9.2.1. The exact solution and differences between the exact and Jacobi solutions after five iterations are shown in Table 9.2.2.

The Gauss-Seidel method for this problem is

$$U_{jk}^{(\nu+1)} = \frac{1}{4}(U_{j+1,k}^{(\nu)} + U_{j-1,k}^{(\nu+1)} + U_{j,k+1}^{(\nu)} + U_{j,k-1}^{(\nu+1)}), \quad \nu = 0, 1, \dots$$

Its solution after five iterations is shown in Table 9.2.3.

The maximum error after five Jacobi iterations is 0.01656 and after five Gauss-Seidel iterations is 0.00146. Thus, as expected, Gauss-Seidel iteration is converging faster than Jacobi iteration

0	0	0	0/1	0	0	0	0/1
0	0	0.25	1	0	0.23438	0.48344	1
0	0.25	0.5	1	0	0.48344	0.73438	1
0/1	1	1	1	0/1	1	1	1

Table 9.2.1: Solution of Example 9.2.4 after one iteration ( $\nu = 0$ , left) and after five iterations ( $\nu = 4$ , right) using Jacobi's method.

0	0	0	0/1	0	0	0	0
0	0.25	0.5	1	0	0.01562	0.01656	0
0	0.5	0.75	1	0	0.01656	0.01562	0
0/1	1	1	1	0	0	0	0

Table 9.2.2: Exact solution of Example 9.2.4 (left) and the errors in the Jacobi solution after five iterations (right).

0	0	0	0/1	0	0	0	0
0	0.24927	0.49963	1	0	0.00073	0.00037	0
0	0.49854	0.74927	1	0	0.00146	0.00073	0
0/1	1	1	1	0	0	0	0

Table 9.2.3: Solution of Example 4 after five iterations ( $\nu = 4$ ) using the Gauss-Seidel method (left) and errors in this solution (right).

*Example 9.2.5.* We'll try to quantify the differences in the convergence rates of Jacobi and Gauss-Seidel iteration for Poisson's equation on a rectangle. Jacobi's method satisfies (9.2.12) and we let  $\mathbf{p}$  be an eigenvector of  $\mathbf{M}_J$  with corresponding eigenvalue  $\mu$ ; thus,

$$\mathbf{M}_J \mathbf{p} = \mu \mathbf{p}. \quad (9.2.17a)$$

Using (9.2.12), we see that the component form of this relation is

$$\begin{aligned} \mu p_{jk} &= \theta_x(p_{j-1,k} + p_{j+1,k}) + \theta_y(p_{j,k-1} + p_{j,k+1}), & j &= 1, 2, \dots, J-1, \\ k &= 1, 2, \dots, K-1, \end{aligned} \quad (9.2.17b)$$

where  $p_{jk}$  is a component of  $\mathbf{p}$ . (The double subscript notation for a vector component is non-standard, but convenient in this case since it corresponds to a position in the finite difference mesh.) One may easily verify that

$$p_{jk} = \sin \frac{mj\pi}{J} \sin \frac{nk\pi}{K} \quad (9.2.17c)$$

and

$$\mu = 1 - 4\theta_x \sin^2 \frac{m\pi}{2J} - 4\theta_y \sin^2 \frac{n\pi}{2K}, \quad m = 1, 2, \dots, J-1, \quad n = 1, 2, \dots, K-1. \quad (9.2.17d)$$

*Remark 1.* The  $(J-1)(K-1)$  eigenvectors and eigenvalues of  $\mathbf{M}_J$  are indexed by  $m$  and  $n$ .

*Remark 2.* The eigenvector  $\mathbf{p}$  is the eigenfunction of the Laplacian sampled at the mesh points  $j = 1, 2, \dots, J-1$ ,  $k = 1, 2, \dots, K-1$ . The eigenvalue  $\mu$  is, however, not an eigenvalue of the Laplacian.

The largest eigenvalue and, hence, the spectral radius of  $\mathbf{M}_J$  may be obtained by setting  $m = n = 1$  in (9.2.17d) to obtain

$$\rho(\mathbf{M}_J) = 1 - 4\theta_x \sin^2 \frac{\pi}{2J} - 4\theta_y \sin^2 \frac{\pi}{2K}. \quad (9.2.17e)$$

In the special case of a square,  $\theta_x = \theta_y = 1/4$  and  $J = K$ ; thus,

$$\rho(\mathbf{M}_J) = 1 - 2 \sin^2 \frac{\pi}{2J}.$$

For large values of  $J$ , we may approximate this as

$$\rho(\mathbf{M}_J) \approx 1 - \frac{\pi^2}{2J^2} = 1 - C\Delta x^2,$$

since  $J = a/\Delta x$  for an  $a \times a$  square region. Thus, the spectral radius approaches unity and the convergence rate slows as  $J$  increases (or as  $\Delta x$  decreases).

In a similar manner, Let  $\mathbf{q}$  and  $\lambda$  be an eigenvector-eigenvalue pair of the Gauss-Seidel iteration matrix  $\mathbf{M}_{GS}$  for Poisson's equation on a rectangle (9.2.16). Thus, using (9.2.15b)

$$\mathbf{M}_{GS}\mathbf{q} = (\mathbf{D} - \mathbf{L})^{-1}\mathbf{U}\mathbf{q} = \lambda\mathbf{q}, \quad (9.2.18a)$$

or, using (9.2.15b), in component form

$$\begin{aligned} \lambda q_{jk} - \lambda(\theta_x q_{j-1,k} + \theta_y q_{j,k-1}) &= (\theta_x q_{j+1,k} + \theta_y q_{j,k+1}), & j &= 1, 2, \dots, J-1, \\ k &= 1, 2, \dots, K-1. & & \end{aligned} \quad (9.2.18b)$$

This problem appears more difficult to analyze than the eigenvalue problem (9.2.17b) for the Jacobi method; however, there is a transformation that simplifies things considerably. Let

$$q_{jk} = \lambda^{(j+k)/2} r_{jk}$$

and substitute this relationship into (9.2.18b) to obtain

$$\lambda^{(j+k+2)/2} r_{jk} - \lambda^{(j+k+1)/2} (\theta_x r_{j-1,k} + \theta_y r_{j,k-1}) = \lambda^{(j+k+1)/2} (\theta_x r_{j+1,k} + \theta_y r_{j,k+1}).$$

Dividing by the common factor yields

$$\lambda^{1/2} r_{jk} = \theta_x (r_{j-1,k} + r_{j+1,k}) + \theta_y (r_{j,k-1} + r_{j,k+1}).$$

This is the same eigenvalue problem as (9.2.17b) for Jacobi's method with  $\mu$  replaced by  $\lambda^{1/2}$ ; thus, using (9.2.17d)

$$\begin{aligned} \lambda = \mu^2 &= \left[ 1 - 4\theta_x \sin^2 \frac{m\pi}{2J} - 4\theta_y \sin^2 \frac{n\pi}{2K} \right]^2, & m &= 1, 2, \dots, J-1, \\ & & n &= 1, 2, \dots, K-1. \end{aligned} \quad (9.2.18c)$$

In particular,

$$\rho(\mathbf{M}_{GS}) = \rho^2(\mathbf{M}_J). \quad (9.2.18d)$$

Thus, according to (9.2.7b), Gauss-Seidel iterations converge twice as fast as Jacobi iterations.

In the special case of Laplace's equation on a square mesh with large  $J$  we obtain the asymptotic approximation

$$\rho(\mathbf{M}_{GS}) \approx 1 - \frac{\pi^2}{J^2}. \quad (9.2.19)$$

The results of Example 9.2.5 generalize as indicated by the following theorem.

**Theorem 9.2.4.** *Suppose that  $\mathbf{A}$  satisfies  $a_{ij}/a_{ii} < 0$ ,  $i \neq j$ , then one and only one of the following conditions can occur:*

1.  $\rho(\mathbf{M}_J) = \rho(\mathbf{M}_{GS}) = 0$ ,
2.  $0 < \rho(\mathbf{M}_{GS}) < \rho(\mathbf{M}_J) < 1$ ,

3.  $\rho(\mathbf{M}_J) = \rho(\mathbf{M}_{GS}) = 1$ , or

4.  $1 < \rho(\mathbf{M}_J) < \rho(\mathbf{M}_{GS})$ .

*Proof.* cf. [6], p. 70. □

In the important Case 2, Gauss-Seidel iterations always converge faster than Jacobi iterations.

### 9.2.2 Successive Over Relaxation

“Relaxation” is a procedure that can accelerate the convergence rate of virtually any iteration. At present, it suits our purposes to apply it to the Gauss-Seidel method. The process begins by using the Gauss-Seidel method (9.2.14) to compute a “provisional” iterate

$$\hat{x}_i^{(\nu+1)} = - \sum_{j=1}^{i-1} \frac{a_{ij}}{a_{ii}} x_j^{(\nu+1)} - \sum_{j=i+1}^n \frac{a_{ij}}{a_{ii}} x_j^{(\nu)} + \frac{b_i}{a_{ii}} \quad (9.2.20a)$$

and concludes with the final iterate

$$x_i^{(\nu+1)} = \omega \hat{x}_i^{(\nu+1)} + (1 - \omega) x_i^{(\nu)}, \quad i = 1, 2, \dots, N. \quad (9.2.20b)$$

The *acceleration parameter*  $\omega$  is to be chosen so that the iteration (9.2.20) converges as fast as possible. In particular, (9.2.20) is called Gauss-Seidel iteration when  $\omega = 1$ , *successive under relaxation* when  $\omega < 1$ , and *successive over relaxation (SOR)* when  $\omega > 1$ . Over relaxation is the important method with elliptic problems.

Using (9.2.8), we can write (9.2.20) in the vector form

$$\mathbf{D}\hat{\mathbf{x}}^{(\nu+1)} = \mathbf{L}x^{(\nu+1)} + \mathbf{U}x^{(\nu)} + \mathbf{b} \quad (9.2.21a)$$

$$\mathbf{x}^{(\nu+1)} = \omega \hat{\mathbf{x}}^{(\nu+1)} + (1 - \omega)\mathbf{x}^{(\nu)}. \quad (9.2.21b)$$

We can further eliminate the provisional iterate and write (9.2.21a, 9.2.21b) in the form of (9.2.2)

$$\mathbf{x}^{(\nu+1)} = \mathbf{M}_\omega \mathbf{x}^{(\nu)} + \hat{\mathbf{b}}_\omega \quad (9.2.21c)$$

with

$$\mathbf{M}_\omega = (\mathbf{D} - \omega\mathbf{L})^{-1}[(1 - \omega)\mathbf{D} + \omega\mathbf{U}], \quad \hat{\mathbf{b}}_\omega = \omega(\mathbf{D} - \omega\mathbf{L})^{-1}\mathbf{b}. \quad (9.2.21d)$$

Our goal is to find the value of  $\omega$  that minimizes  $\rho(\mathbf{M}_\omega)$  and, hence, maximizes the convergence rate. There is a wealth of theory on this subject and let us begin with some preliminary considerations.

**Definition 9.2.3.** A matrix  $\mathbf{A}$  is two cyclic if there is a permutation of its rows and columns that reduce it to the form

$$\begin{bmatrix} \mathbf{D}_1 & \mathbf{F} \\ \mathbf{G} & \mathbf{D}_2 \end{bmatrix}$$

where  $\mathbf{D}_1$  and  $\mathbf{D}_2$  are diagonal.

**Definition 9.2.4.** A matrix  $\mathbf{A}$  is weakly two cyclic if  $\mathbf{D}_1$  and  $\mathbf{D}_2$  are zero.

*Example 9.2.6.* The matrix shown in Figure 9.2.2 is two cyclic as revealed by an interchange of its second and third rows and columns.

$$\begin{bmatrix} 1 & c & 0 \\ a & 1 & c \\ 0 & a & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & c & 0 \\ 0 & a & 1 \\ a & 1 & c \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & c \\ 0 & 1 & a \\ a & c & 1 \end{bmatrix}$$

Figure 9.2.2: Matrix (left) whose second and third rows are interchanged (center) and whose second and third columns are interchanged to obtain a two-cyclic form (right).

*Example 9.2.7.* Consider the Laplacian operator on a  $4 \times 4$  grid as shown in Figure 9.2.3. Instead of ordering the equations and unknowns by rows, order them in “checkerboard” or “red-black” fashion by listing unknowns and equations at every other point. The resulting matrix has the two-cyclic form

$$\begin{bmatrix} \times & & & & \times & \times & & & \\ & \times & & & \times & & \times & & \\ & & \times & & \times & \times & \times & \times & \\ & & & \times & & \times & & \times & \\ \times & \times & \times & & \times & & & & \\ \times & & \times & \times & & \times & & & \\ & \times & \times & & \times & & \times & & \\ & & \times & \times & \times & & & \times & \end{bmatrix}.$$

4	9	5	
7	3	8	
1	6	2	

Figure 9.2.3: Red-black ordering of the Laplacian operator on a  $4 \times 4$  square mesh

**Definition 9.2.5.** A two-cyclic matrix of the form of (9.2.8) is consistently ordered if the eigenvalues of

$$\mathbf{D}^{-1}(\alpha\mathbf{L} + \frac{1}{\alpha}\mathbf{U})$$

are independent of  $\alpha$  for all real  $\alpha \neq 0$ .

We're now ready to search for the optimal choice of  $\omega$ .

**Theorem 9.2.5.** If a matrix  $\mathbf{A}$  is consistently ordered then the eigenvalues  $\lambda$  of  $\mathbf{M}_\omega$  and  $\mu$  of  $\mathbf{M}_J$  are related by

$$\mu = \frac{\lambda + \omega - 1}{\lambda^{1/2}\omega}. \quad (9.2.22)$$

*Proof.* From (9.2.21d), we see that the eigenvalues of  $\mathbf{M}_\omega$  satisfy

$$(\mathbf{D} - \omega\mathbf{L})^{-1}[(1 - \omega)\mathbf{D} + \omega\mathbf{U}]\mathbf{q} = \lambda\mathbf{q}$$

where  $\mathbf{q}$  is the eigenvector of  $\mathbf{M}_\omega$  corresponding to  $\lambda$ . Multiplying by  $(\mathbf{D} - \omega\mathbf{L})$ , we have

$$[(1 - \omega)\mathbf{D} + \omega\mathbf{U} - \lambda(\mathbf{D} - \omega\mathbf{L})]\mathbf{q} = \mathbf{0}.$$

Multiplying by  $\mathbf{D}^{-1}/\omega$

$$[\mathbf{D}^{-1}(\mathbf{U} + \lambda\mathbf{L}) - \frac{\lambda + \omega - 1}{\omega}\mathbf{I}]\mathbf{q} = \mathbf{0}.$$

Finally, multiplying by  $\lambda^{-1/2}$  yields

$$[\mathbf{D}^{-1}(\lambda^{1/2}\mathbf{L} + \lambda^{-1/2}\mathbf{U}) - \mu\mathbf{I}]\mathbf{q} = \mathbf{0}$$

where  $\mu$  satisfies (9.2.22). Thus  $\mu$  is an eigenvalue of

$$\mathbf{D}^{-1}(\lambda^{1/2}\mathbf{L} + \lambda^{-1/2}\mathbf{U}).$$

If  $\mathbf{A}$  is consistently ordered then the eigenvalues of this matrix are independent of the parameter  $\lambda^{1/2}$ . Thus, we can choose any convenient value of  $\lambda$  to find the eigenvalues  $\mu$ . In particular, if we choose  $\lambda = 1$  then  $\mu$  is an eigenvalue of  $\mathbf{M}_J$  (cf. (9.2.10b)).  $\square$

*Remark 3.* Setting  $\omega = 1$  for the Gauss-Seidel method and using (9.2.22), we see that  $\mu = \lambda^{1/2}$ , confirming the relationship between the eigenvalues of  $\mathbf{M}_J$  and  $\mathbf{M}_{GS}$  that we found in Example 9.2.5 for the discrete Laplacian.

*Remark 4.* The transformation used in Example 9.2.5 could have also been used to prove this theorem for the discrete Laplacian operator.

Let us assume that the eigenvalues  $\mu$  of  $\mathbf{M}_J$  are real. (It suffices to assume that  $\mathbf{A}$  is symmetric.) Let us also assume that  $\rho(\mathbf{M}_J) < 1$ . Then, using Theorem 9.2.4,  $\rho(\mathbf{M}_{GS}) < 1$ . Let

$$f(\lambda, \omega) = \frac{\lambda + \omega - 1}{\omega}, \quad g(\lambda, \mu) = \mu\lambda^{1/2}. \quad (9.2.23)$$

We sketch  $f(\lambda, \omega)$  and  $g(\lambda, \mu)$  as functions of  $\lambda$  in Figure 9.2.4. Both halves of  $g(\lambda, \mu)$  are shown since the eigenvalues of  $\mathbf{M}_J$  occur in pairs. Thus, if  $\mu$  is an eigenvalue of  $\mathbf{M}_J$ , so is  $-\mu$ . This may be shown for Laplace's equation using the results of Example 9.2.5, but we won't do it here.

Let's list several properties of  $f(\lambda, \omega)$  and  $g(\lambda, \mu)$  that can be discerned from (9.2.23) and Figure 9.2.23.

1.  $f(1, \omega) = 1, \forall \omega$ .
2. The function  $g(\lambda, \mu)$  increases linearly with  $|\mu|$ . Its largest amplitude occurs when  $\mu = \rho(\mathbf{M}_J)$ .
3. For fixed  $\mu$  and  $\omega$ , the eigenvalues of  $\mathbf{M}_\omega$  are given by the values of  $\lambda$  where  $f(\lambda, \omega) = g(\lambda, \mu)$  (cf. (9.2.22)). With  $\mu = \rho(\mathbf{M}_J)$ , these eigenvalues are shown as points  $A$  and  $B$  on Figure 9.2.4. The larger value of  $\lambda$  at the points labeled  $A$  corresponds to  $\rho(\mathbf{M}_\omega)$  (Item 2).

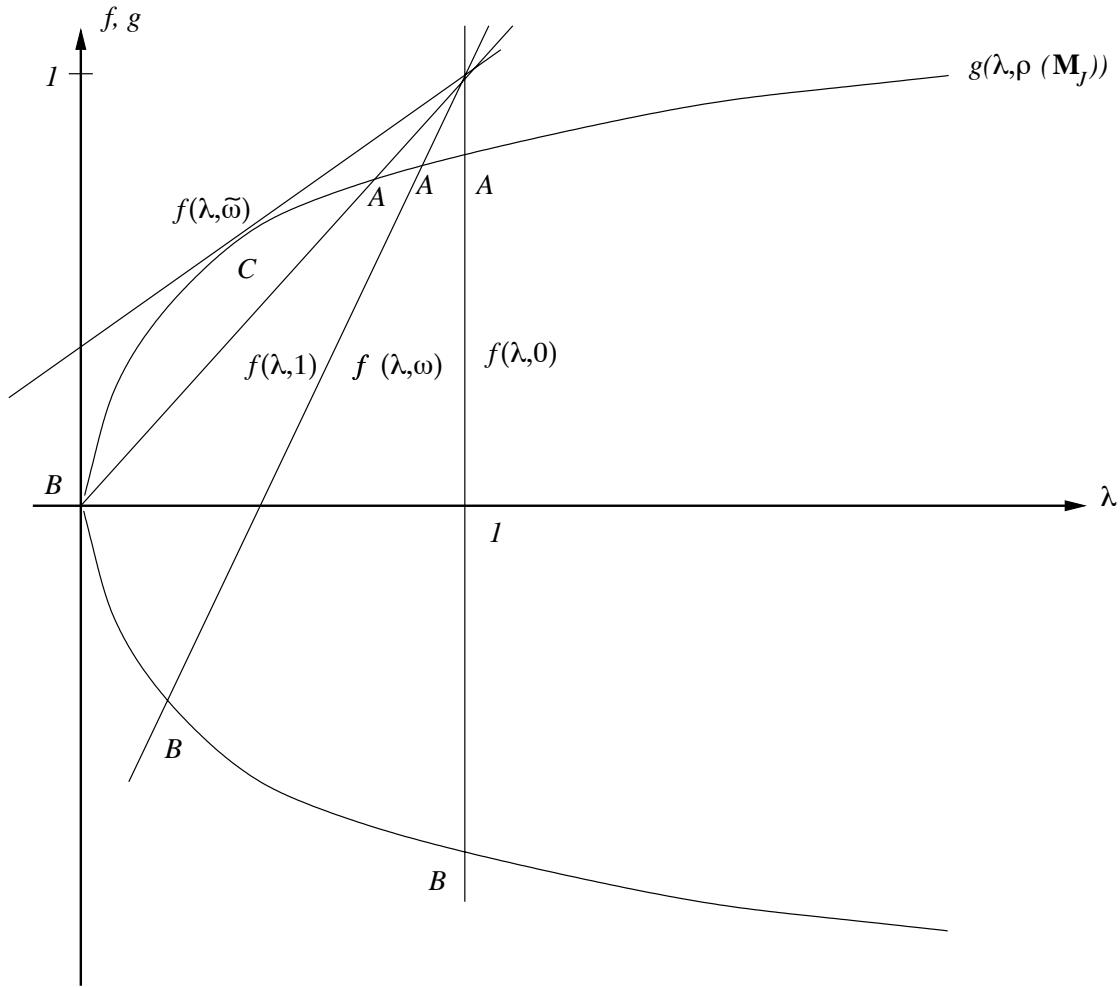


Figure 9.2.4: Functions  $f(\lambda, \omega)$  and  $g(\lambda, \rho(\mathbf{M}_J))$  vs.  $\lambda$ .

4. Setting  $\omega = 0$  gives  $f(\lambda, 0)$  as the line  $\lambda = 1$ . If  $\omega < 0$ , then the values of  $\lambda$  at the intersection points (not shown in Figure 9.2.4) would exceed unity, and the iteration (9.2.21) would diverge.
5. Setting  $\omega = 1$  in (9.2.23) gives  $f(\lambda, 1) = \lambda$ , which is the Gauss-Seidel method.
6. As seen in Figure 9.2.4, the maximum eigenvalue  $\lambda$  of  $\mathbf{M}_\omega$  can be reduced further by choosing  $\omega > 1$ . The minimum real solution for  $\lambda$  occurs at  $\omega = \tilde{\omega}$  when  $f(\lambda, \tilde{\omega})$  is tangent (at point  $C$ ) to  $g(\lambda, \rho(\mathbf{M}_J))$ .

Let us rewrite (9.2.22) as

$$\lambda - \omega\mu\lambda^{1/2} + \omega - 1 = 0.$$

This is a quadratic equation in  $\lambda^{1/2}$ ; thus,

$$\lambda = \left[ \omega\mu/2 \pm \sqrt{(\omega\mu/2)^2 + 1 - \omega} \right]^2. \quad (9.2.24a)$$

The point of tangency ( $C$ ) occurs at  $\tilde{\omega}$  when

$$(\tilde{\omega}\mu/2)^2 + 1 - \tilde{\omega} = 0$$

or

$$\tilde{\omega}_{\pm} = \frac{2}{1 \pm \sqrt{1 - \mu^2}}.$$

For  $0 \leq \mu \leq 1$ , we see that  $1 \leq \tilde{\omega}_+ \leq 2$  and  $2 \leq \tilde{\omega}_- < \infty$ . We'll show in a moment that the interesting range of  $\omega$  is  $[0, 2]$ ; thus, the appropriate value of  $\tilde{\omega}$  is  $\tilde{\omega}_+$ . We'll drop the + subscript and simply let

$$\tilde{\omega} = \frac{2}{1 + \sqrt{1 - \mu^2}}. \quad (9.2.24b)$$

Substituting (9.2.24b) into (9.2.24a) yields

$$\tilde{\lambda} = \lambda(\tilde{\omega}) = (\tilde{\omega}\mu/2)^2 = \tilde{\omega} - 1. \quad (9.2.24c)$$

Values of  $\omega < \tilde{\omega}$  produce real values of  $\lambda$  while values of  $\omega > \tilde{\omega}$  yield complex values of  $\lambda$ . In this latter case, we may use (9.2.24a) to show that

$$|\lambda| = \omega - 1, \quad \omega > \tilde{\omega}.$$

We're now able to state our main result.

**Theorem 9.2.6.** *Let  $\mathbf{A}$  be a symmetric and consistently-ordered matrix, then the optimal relaxation parameter satisfies*

$$\omega_{OPT} = \frac{2}{1 + \sqrt{1 - \rho^2(\mathbf{M}_J)}} \quad (9.2.25a)$$

and

$$\rho(\mathbf{M}_{\omega}) = \begin{cases} \left[ \omega\rho(\mathbf{M}_J)/2 + \sqrt{(\omega\rho(\mathbf{M}_J)/2)^2 + 1 - \omega} \right]^2, & \text{if } 0 \leq \omega \leq \omega_{OPT} \\ \omega - 1, & \text{if } \omega_{OPT} < \omega \end{cases}. \quad (9.2.25b)$$

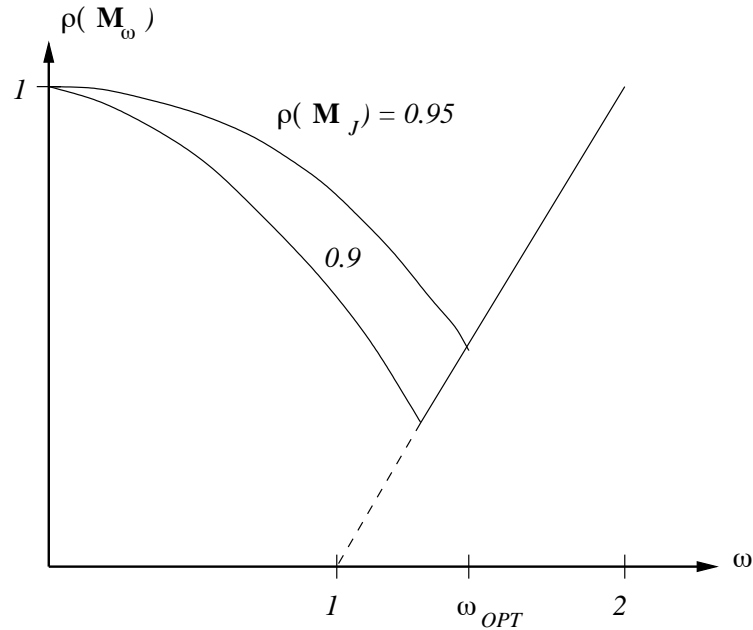


Figure 9.2.5: Spectral radius  $\rho(\mathbf{M}_\omega)$  as a function of  $\omega$  for SOR iteration.

*Proof.* With the various symmetries (*cf.* Strikwerda [5], Section 13.3), it suffices to consider positive values of  $\mu$ . With this choice and our interest in calculating the spectral radius of  $\mathbf{M}_\omega$ , we clearly want the positive sign in (9.2.24a). Thus, in summary, we have

1.  $\lambda = 1$  when  $\omega = 0$ ,
2.  $\lambda = \left[ \omega\mu/2 + \sqrt{(\omega\mu/2)^2 + 1 - \omega} \right]^2$ ,  $0 < \omega \leq \tilde{\omega}$ , and
3.  $|\lambda| = \omega - 1$ ,  $\tilde{\omega} < \omega$ .

The values of  $\lambda$  increase with  $\mu$ ; thus, the spectral radius  $\rho(\mathbf{M}_\omega)$  occurs when  $\mu = \rho(\mathbf{M}_J)$ , as given by (9.2.25b). The minimum value of  $\rho(\mathbf{M}_\omega)$  occurs at  $\tilde{\omega}$  corresponding to  $\mu = \rho(\mathbf{M}_J)$ , as given by (9.2.25a). The spectral radius  $\rho(\mathbf{M}_\omega)$  is displayed as a function of  $\omega$  in Figure 9.2.5. □

Several corollaries follow from Theorem 9.2.6 and we'll list one.

**Corollary 9.2.1.** *Under the conditions of Theorem 9.2.6, the SOR method converges for  $\omega \in (0, 2)$  when  $\rho(\mathbf{M}_J) < 1$ .*

*Proof.* From (9.2.25)

$$\rho(\mathbf{M}_{\omega_{OPT}}) = \omega_{OPT} - 1 = \frac{2}{1 + \sqrt{1 - \rho^2(\mathbf{M}_J)}} - 1.$$

Thus,  $0 < \rho(\mathbf{M}_{\omega_{OPT}}) < 1$  when  $0 < \rho(\mathbf{M}_J) < 1$ .

Additional inspection of (9.2.25b) reveals that  $d\rho(\mathbf{M}_\omega)/d\omega$  is non-positive when  $\omega < \omega_{OPT}$  and unity when  $\omega > \omega_{OPT}$  (Figure 9.2.5). Since  $\rho(\mathbf{M}_\omega) = 1$  at  $\omega = 0, 2$ , we conclude that  $\rho(\mathbf{M}_\omega) < 1$  for  $0 < \omega < 2$ .  $\square$

An examination of (9.2.25) and Figure 9.2.5 reveals that  $\omega_{OPT} \in (1, 2)$ . Increasing  $\rho(\mathbf{M}_J)$  increases  $\omega_{OPT}$  and, in particular,  $\omega_{OPT} \rightarrow 2$  as  $\rho(\mathbf{M}_J) \rightarrow 1$ . From Figure 9.2.5, we also see that overestimating  $\omega_{OPT}$  by a given amount increases  $\rho(\mathbf{M}_\omega)$  less than underestimating it by the same amount.

*Example 9.2.8.* Let us solve the problem of Example 9.2.4 using SOR iteration. With  $J = K = 3$  and  $\theta_x = \theta_y = 1/4$ , we obtain the Gauss-Seidel method (9.2.16) as

$$\hat{U}_{jk}^{(\nu+1)} = \frac{1}{4}(U_{j+1,k}^{(\nu)} + U_{j-1,k}^{(\nu+1)} + U_{j,k+1}^{(\nu)} + U_{j,k-1}^{(\nu+1)}).$$

With relaxation, we compute

$$U_{jk}^{(\nu+1)} = \omega \hat{U}_{jk}^{(\nu+1)} + (1 - \omega)U_{jk}^{(\nu)}.$$

We'll explicitly eliminate the intermediate variable to obtain the SOR method as

$$U_{jk}^{(\nu+1)} = \frac{\omega}{4}(U_{j+1,k}^{(\nu)} + U_{j-1,k}^{(\nu+1)} + U_{j,k+1}^{(\nu)} + U_{j,k-1}^{(\nu+1)}) + (1 - \omega)U_{jk}^{(\nu)}.$$

Using (9.2.17e) with  $J = K = 3$ , the spectral radius of the Jacobi method is

$$\rho(\mathbf{M}_J) = 1 - \sin^2 \frac{\pi}{6} - \sin^2 \frac{\pi}{6} = \frac{1}{2};$$

thus, using (9.2.25a)

$$\omega_{OPT} = \frac{2}{1 + \sqrt{1 - 1/4}} \approx 1.07.$$

The SOR solution and errors after five iterations are shown in Table 9.2.4. After five iterations, the percentage errors at point  $(1, 1)$  are 1.2, 0.29, and 0.014, respectively, for the Jacobi, Gauss-Seidel, and SOR methods (*cf.* Example 9.2.4).

0	0	0	0/1	0	0	0	0
0	0.24997	0.49999	1	0	0.00003	0.00001	0
0	0.49993	0.74998	1	0	0.00007	0.00002	0
0/1	1	1	1	0	0	0	0

Table 9.2.4: Solution of Example 9.2.8 after five iterations ( $\nu = 4$ ) using the SOR method with  $\omega = 1.07$  (left) and errors in this solution (right).

*Example 9.2.9.* Let us examine the convergence rate of SOR a bit more closely for Laplace's equation. Using (9.2.17e), the spectral radius of Jacobi's method on a square is

$$\rho(\mathbf{M}_J) = 1 - 2 \sin^2 \frac{\pi}{2J} = \cos \frac{\pi}{J}.$$

Using (9.2.25a)

$$\omega_{OPT} = \frac{2}{1 + \sqrt{1 - \cos^2 \pi/J}} = \frac{2}{1 + \sin \pi/J}.$$

Now, using (9.2.25b)

$$\rho(\mathbf{M}_{\omega_{OPT}}) = \omega_{OPT} - 1 = \frac{1 - \sin \pi/J}{1 + \sin \pi/J}$$

or, for large values of  $J$ ,

$$\rho(\mathbf{M}_{\omega_{OPT}}) \approx 1 - \frac{2\pi}{J}.$$

Recall (Example 9.2.5) that the spectral radius of Jacobi and Gauss-Seidel iteration under the same conditions is  $1 - O(1/J^2)$ . Thus, SOR iteration is considerably better. We'll emphasize this by computing the convergence rate according to (9.2.7b). For the Jacobi method, we find

$$R_\nu(\mathbf{M}_J) \leq -\ln \rho(\mathbf{M}_J) \approx -\ln\left(1 - \frac{\pi^2}{2J^2}\right) \approx \frac{\pi^2}{2J^2}.$$

Similarly, for the Gauss-Seidel method, we have

$$R_\nu(\mathbf{M}_{GS}) \approx \frac{\pi^2}{J^2},$$

and for SOR, we have

$$R_\nu(\mathbf{M}_{\omega_{OPT}}) \approx \frac{2\pi}{J}.$$

Thus, typically, the Jacobi or Gauss-Seidel methods would require  $O(J^2)$  iterations to obtain an answer having a specified accuracy while the SOR would obtain the same accuracy in only  $O(J)$  iterations.

The optimal relaxation parameter is not known for realistic elliptic problems because the eigenvalues of  $\mathbf{M}_J$  are typically unavailable. Strikwerda [5], Section 13.3, describes a way of calculating approximate values of  $\omega_{OPT}$ . The optimal relaxation parameter for many elliptic problems is close to 2 and may be approximated by an expression of the form

$$\omega_{OPT} = \frac{2}{1 + Ch} \quad (9.2.26)$$

where  $h$  is some measure of the grid spacing, *e.g.*,  $h = \max(\Delta x, \Delta y)$ . The value of the constant  $C$  can be determined by calculating  $\omega_{OPT}$  on some coarse grids and then extrapolating to finer grids. The values of  $\omega_{OPT}$  on a coarse grid is determined experimentally by making several computations on the grid with different values of  $\omega$ . The value that produces the fewest iterations for a given level of accuracy is assumed to be  $\omega_{OPT}$ .

Some common variations of SOR iteration follow:

1. *Red-Black (Checkerboard) ordering.* We presented this ordering in Example 9.2.7. As usual, let a point in a rectangular mesh be denoted as  $(j, k)$ . With red-black ordering, we number all equations and unknowns at, *e.g.*, odd values of  $j + k$  before those with even values of  $j + k$  (Figure 9.2.3). Recall, that this gave us a system of the form

$$\mathbf{Ax} = \begin{bmatrix} \mathbf{D}_1 & \mathbf{F} \\ \mathbf{G} & \mathbf{D}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

where  $\mathbf{D}_1$  and  $\mathbf{D}_2$  are diagonal and  $\mathbf{x}_1$  and  $\mathbf{x}_2$  correspond to unknowns at odd- and even-numbered points, respectively. SOR iteration is performed first on the odd points and then on the even points. Note that the updating of an unknown at each odd point is independent of that at any other odd point; hence, they may be done in parallel without a need for synchronization. Similarly, unknowns at all even points may be updated in parallel.

2. *Symmetric ordering (SSOR).* Generally, the matrix  $\mathbf{M}_\omega$  is not symmetric even when the original matrix  $\mathbf{A}$  is. There are instances when it is important to maintain symmetry, *e.g.*, when using SOR iteration as a preconditioning for the conjugate gradient method (Section 9.3). A symmetric iteration matrix can be obtained by

performing a standard SOR sweep with, say, row ordering followed by one with the reverse of this ordering.

3. *Line or block procedures.* Order the unknowns by rows, but gather all of the unknowns in a row into a vector to obtain a “block tridiagonal” system. For Poisson’s equation, this system was given as (8.1.4). It is partially reproduced here for convenience as

$$\mathbf{Ax} = \begin{bmatrix} \mathbf{C}_1 & \mathbf{D}_1 & & & \\ \mathbf{D}_2 & \mathbf{C}_2 & \mathbf{D}_2 & & \\ & & \ddots & & \\ & & & \mathbf{D}_{K-1} & \mathbf{C}_{K-1} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_{K-1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_{K-1} \end{bmatrix},$$

The matrices  $\mathbf{C}_k$ , *etc.* were defined by (8.1.4c-8.1.4d). The SOR procedure is applied to an entire row, *i.e.*, we compute

$$\begin{aligned} \mathbf{C}_k \hat{\mathbf{x}}_k^{(\nu+1)} &= -\mathbf{D}_k \mathbf{x}_{k-1}^{(\nu+1)} - \mathbf{D}_k \mathbf{x}_{k+1}^{(\nu)} + \mathbf{b}_k, \\ \mathbf{x}_k^{(\nu+1)} &= \omega \hat{\mathbf{x}}_k^{(\nu+1)} + (1 - \omega) \mathbf{x}_k^{(\nu)}, \quad k = 1, 2, \dots, K - 1, \quad \nu = 0, 1, \dots \end{aligned}$$

Thus, we have to solve a tridiagonal system at each step of the process. This procedure converges faster than point SOR iteration by a factor of  $\sqrt{2}$ .

4. *Alternating direction implicit (ADI) methods.* By considering an elliptic problem as the steady state limit of a transient parabolic problem, we can use some methods for time-dependent problems to solve them. In particular, the ADI method (*cf.* Section 5.2) has been adapted to the solution of elliptic problems. The goal, when using this approach, is to select the “time step” so that the ADI scheme converges to steady state as fast as possible. The ADI method with a single acceleration parameter (artificial time step) has the same convergence rate as SOR. Further acceleration is possible by choosing a sequence of artificial time steps, changing them after each predictor-corrector sweep, and applying them cyclically. Wachspress [7] has shown how to select nearly optimal acceleration parameters.

The results shown in Table 9.2.5 summarize the convergence rates of the methods that we have studied in this section. They are all obtained by solving a Dirichlet problem on a

square mesh with uniform spacing  $h = \Delta x = \Delta y$ . The convergence rates of all methods decline as the mesh spacing decreases. Degradation in performance is least with the ADI method; however, computing optimal parameters can be problematical in realistic situations.

Method	Conv. Rate
Jacobi	$h^2$
Gauss-Seidel	$2h^2$
SOR (with $\omega_{OPT}$ )	$2h$
ADI ( with $m$ parameters)	$\frac{8}{m}(\frac{h}{2})^{1/m}$

Table 9.2.5: Convergence rates for various iterative methods as a function of mesh spacing  $h$  for a Dirichlet problem on a square.

### Problems

1. Consider a problem for Laplace's equation

$$u_{xx} + u_{yy} = 0, \quad (x, y) \in \Omega,$$

where  $\Omega$  is the region between a  $4 \times 4$  square and a concentric  $2 \times 2$  square (Figure 9.2.6). The Dirichlet boundary conditions are  $u = 1$  on the outside of the  $4 \times 4$  square and  $u = 0$  on the edge of the  $2 \times 2$  square. Due to symmetry, this problem need only be solved on the one octant shown on the right of Figure 9.2.6. The subscript  $\mathbf{n}$  denotes differentiation in the outer normal direction.

- 1.1. Construct a discrete approximation of the above problem on the region shown on the right of Figure 9.2.6. Use the five point difference approximation for Laplace's equation and, for simplicity, assume that the mesh spacing in the  $x$  and  $y$  directions is the same, say,  $\Delta x = \Delta y = 1/N$ . Take appropriate steps to ensure that the finite difference approximations at the symmetry boundary and the interior have  $O(1/N^2)$  accuracy.
- 1.2. For the special discretization when  $\Delta x = 1/2$  the discrete problem has only four unknowns. Write down an SOR procedure for determining these unknowns. Calculate the Jacobi iteration matrix  $\mathbf{M}_J$ . Find an expression for

the spectral radius  $\rho(\mathbf{M}_{SOR})$  of the SOR matrix. Plot  $\rho(\mathbf{M}_{SOR})$  and determine the optimal relaxation parameter  $\omega$ . (This problem should be done symbolically.)

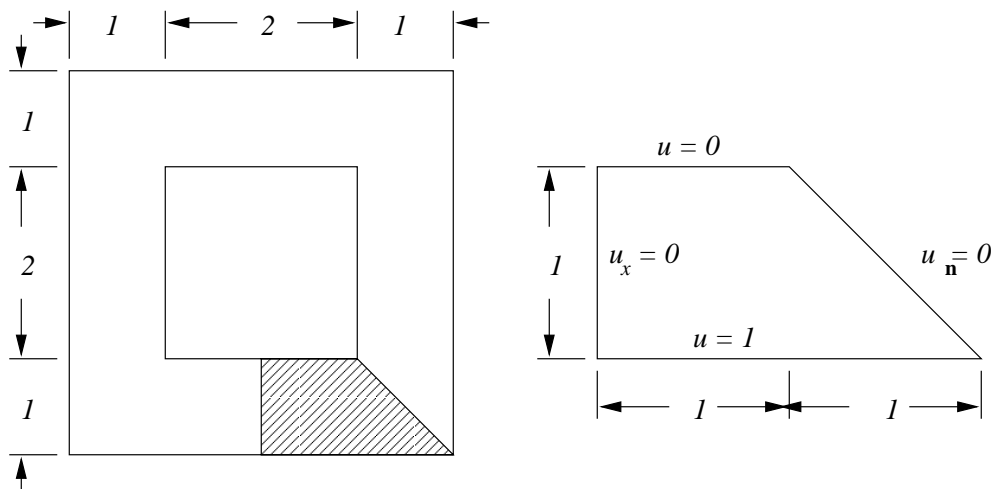


Figure 9.2.6: Domain for Problem 1 (left). Due to symmetry, the problem need only be solved on the octant shown on the right.

### 9.3 Conjugate Gradient Methods

The fixed-point iterative methods of the previous section deteriorate in performance as the dimension  $N$  of the linear system increases. The faster SOR and ADI techniques depend on acceleration parameters that may be difficult to estimate. We seek to overcome these deficiencies without raising the storage requirements to the level of a direct method. Solving the linear system (9.1.1) when  $\mathbf{A}$  is symmetric positive definite matrix is equivalent to minimizing the quadratic functional

$$E(\mathbf{y}) = \frac{1}{2}\mathbf{y}^T \mathbf{A} \mathbf{y} - \mathbf{b}^T \mathbf{y}. \quad (9.3.1)$$

The necessary condition for a minimum,

$$E'(\mathbf{y}) = \mathbf{A} \mathbf{y} - \mathbf{b} = \mathbf{0}, \quad (9.3.2a)$$

implies that  $\mathbf{y} = \mathbf{x}$ , the solution of (9.1.1).

If we define the residual

$$\mathbf{r}(\mathbf{y}) = \mathbf{b} - \mathbf{A}\mathbf{y} \quad (9.3.2b)$$

then (9.3.2a) may be written as

$$E'(\mathbf{y}) = -\mathbf{r}(\mathbf{y}). \quad (9.3.2c)$$

The level surfaces  $E(\mathbf{y}) = C$  (a constant) of (9.3.1) are ellipsoids in  $\Re^N$  with a common center at  $\mathbf{x}$ . Since the gradient of a function is in the direction of steepest increase, to minimize a  $E(\mathbf{y})$  at a point  $\mathbf{x}^{(0)}$ , we could move in a direction opposite to the gradient of the level surface through  $\mathbf{x}^{(0)}$ . From (9.3.2c), the gradient at  $\mathbf{x}^{(0)}$  is

$$E'(\mathbf{x}^{(0)}) = -\mathbf{r}(\mathbf{x}^{(0)}) = -\mathbf{r}^{(0)}.$$

Let our subsequent guess  $\mathbf{x}^{(1)}$  for the minimum  $\mathbf{x}$  be

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} + \alpha_0 \mathbf{r}^{(0)},$$

and let us calculate the distance  $\alpha_0$  moved in the negative gradient direction  $\mathbf{r}^{(0)}$  so as to minimize  $E(\mathbf{x}^{(1)})$ . Using (9.3.1), we have

$$E(\mathbf{x}^{(1)}) = E(\mathbf{x}^{(0)} + \alpha_0 \mathbf{r}^{(0)}) = \frac{1}{2}(\mathbf{x}^{(0)} + \alpha_0 \mathbf{r}^{(0)})^T \mathbf{A}(\mathbf{x}^{(0)} + \alpha_0 \mathbf{r}^{(0)}) - \mathbf{b}^T(\mathbf{x}^{(0)} + \alpha_0 \mathbf{r}^{(0)}).$$

Differentiating with respect to  $\alpha_0$

$$\frac{d}{d\alpha_0} E(\mathbf{x}^{(0)} + \alpha_0 \mathbf{r}^{(0)}) = [(\mathbf{x}^{(0)})^T \mathbf{A} - \mathbf{b}^T + \alpha_0 (\mathbf{r}^{(0)})^T \mathbf{A}] \mathbf{r}^{(0)} = 0.$$

Using (9.3.2b)

$$\alpha_0 = \frac{(\mathbf{r}^{(0)})^T \mathbf{r}^{(0)}}{(\mathbf{r}^{(0)})^T \mathbf{A} \mathbf{r}^{(0)}}.$$

With subsequent iterates computed in the same manner, the process is called *the method of steepest descent*. A pseudocode algorithm of the method appears in Figure 9.3.1. Some comments on the procedure and method follow.

1. The calculation of  $\mathbf{r}^{(\nu+1)}$  shown in the algorithm follows definition (9.3.2b); thus,

$$\mathbf{r}^{(\nu+1)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(\nu+1)} = \mathbf{b} - \mathbf{A}(\mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{r}^{(\nu)}) = \mathbf{r}^{(\nu)} - \alpha_\nu \mathbf{A}\mathbf{r}^{(\nu)}. \quad (9.3.3)$$

Formula (9.3.3) is less susceptible to the accumulation of round-off error than direct computation using (9.3.2b).

```

procedure steepest_descent
   $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ 
   $\nu = 0$ 
  while not converged do
     $\alpha_\nu = (\mathbf{r}^{(\nu)})^T \mathbf{r}^{(\nu)} / (\mathbf{r}^{(\nu)})^T \mathbf{A} \mathbf{r}^{(\nu)}$ 
     $\mathbf{x}^{(\nu+1)} = \mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{r}^{(\nu)}$ 
     $\mathbf{r}^{(\nu+1)} = \mathbf{r}^{(\nu)} - \alpha_\nu \mathbf{A} \mathbf{r}^{(\nu)}$ 
     $\nu = \nu + 1$ 
  end while

```

Figure 9.3.1: A steepest-descent algorithm.

- The algorithm only has one matrix multiplication and two vector multiplications per step. When solving partial differential equations, it is not necessary to store the matrix  $\mathbf{A}$ . The product  $\mathbf{A}\mathbf{r}^{(\nu)}$  can be obtained directly from the difference scheme. For example, when solving a problem for Poisson's equation using centered differences (8.1.2), we could compute  $\mathbf{A}\mathbf{r}^{(\nu)}$  at grid point  $(j, k)$  as

$$\mathbf{A}\mathbf{r}_{jk}^{(\nu)} = r_{jk}^{(\nu)} - \theta_x(r_{j+1,k}^{(\nu)} + r_{j-1,k}^{(\nu)}) - \theta_y(r_{j,k+1}^{(\nu)} + r_{j,k-1}^{(\nu)}).$$

The matrix  $\mathbf{A}$  is not stored and the vector  $\mathbf{r}^{(\nu)}$  is stored in the mesh coordinates.

- Using (9.3.3) and the definition of  $\alpha_\nu$  given in the algorithm of Figure 9.3.1, we have

$$(\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu)} = (\mathbf{r}^{(\nu)})^T \mathbf{r}^{(\nu)} - \alpha_\nu (\mathbf{r}^{(\nu)})^T \mathbf{A} \mathbf{r}^{(\nu)} = 0.$$

Thus, the search directions  $\mathbf{r}^{(k)}$ ,  $k = 0, 1, \dots$ , are “orthogonal” in the sense that

$$(\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu)} = 0.$$

- The steepest descent method converges from any initial guess  $\mathbf{x}^{(0)}$  when  $\mathbf{A}$  and  $\mathbf{A}^T \mathbf{A}^{-1}$  are positive definite; thus, symmetry of  $\mathbf{A}$  is not necessary ([5], Section 14.1).
- Let us introduce the *strain energy norm*

$$\|\mathbf{x}\|_{\mathbf{A}}^2 := \mathbf{x}^T \mathbf{A} \mathbf{x}. \tag{9.3.4a}$$

Then ([4], Section 5.3)

$$\|\mathbf{x}^{(\nu+1)} - \mathbf{x}\|_{\mathbf{A}}^2 \leq \left(1 - \frac{1}{\kappa_2(\mathbf{A})}\right) \|\mathbf{x}^{(\nu)} - \mathbf{x}\|_{\mathbf{A}}^2 \quad (9.3.4b)$$

where  $\kappa_2$  is the *condition number* of  $\mathbf{A}$  in the  $\mathcal{L}^2$  norm as given by

$$\kappa_2(\mathbf{A}) = \lambda_{max}/\lambda_{min}, \quad (9.3.4c)$$

and  $\lambda_{max}$  and  $\lambda_{min}$  are the maximum and minimum eigenvalues of  $\mathbf{A}$ . Convergence is slow when  $\kappa(\mathbf{A})$  is large and  $\mathbf{A}$  is said to be ill conditioned in this case. When this occurs, the level surfaces of  $E(\mathbf{y})$  are often elongated ellipses with the minimum  $\mathbf{x}$  lying at the bottom of a narrow valley with steep walls. Successive iterates tend to wander back and forth across the valley making little progress towards the minimum.

*Example 9.3.1.* Let's solve a simple problem with

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 1 & 10 \end{bmatrix}$$

to illustrate some geometrical aspects of the method of steepest descent. With  $\mathbf{b} = \mathbf{0}$ , the minimum of (9.3.1) is at the origin. We select the initial guess

$$\mathbf{x}^{(0)} = [0.1972, 0.4443]^T$$

to be such that  $E(\mathbf{x}^{(0)}) = 1$  (Figure 9.3.2). Since  $\mathbf{b} = \mathbf{0}$ , we obtain

$$\mathbf{r}^{(0)} = [-0.8387, -4.6406], \quad \alpha_0 = 0.0990, \quad \mathbf{x}^{(1)} = [0.1141, -0.0153]^T.$$

The second iteration produces

$$\mathbf{r}^{(1)} = [-0.2130, 0.0385]^T, \quad \alpha_1 = 0.5255, \quad \mathbf{x}^{(2)} = [0.0022, 0.0050]^T.$$

The iteration seems to be converging to the minimum at the origin. These two iterates and the level surfaces  $E(\mathbf{y}) = 1, 0.75, 0.5, 0.25$  are shown in Figure 9.3.2. The initial iteration proceeds “downhill” in a direction opposite to the gradient at  $\mathbf{x}^{(0)}$  until a local minimum is reached. The second iterate proceeds from there.

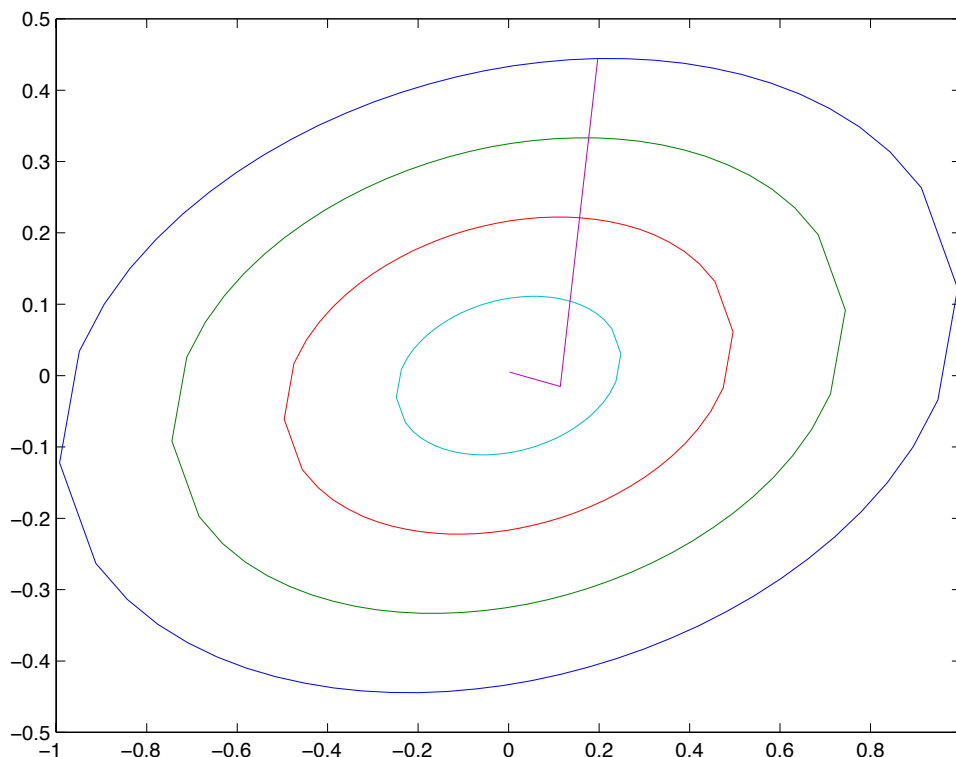


Figure 9.3.2: Convergence of the method of steepest descent for the problem of Example 9.3.1. Ellipses have level values of 1, 0.75, 0.5, and 0.25 (outer to inner). The first two iterations are shown to be converging to the solution at  $(0, 0)$ .

### 9.3.1 The Conjugate Gradient Method

The remedy for the slow convergence encountered with the method of steepest descent is to choose other search directions. With the conjugate gradient method, we choose

$$\mathbf{x}^{(\nu+1)} = \mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{p}^{(\nu)} \quad (9.3.5a)$$

with

$$\mathbf{p}^{(\nu)} = \mathbf{r}^{(\nu)} + \gamma_\nu (\mathbf{x}^{(\nu)} - \mathbf{x}^{(\nu-1)}). \quad (9.3.5b)$$

Thus, the new search direction is a linear combination of the steepest descent search direction and the previous “correction”  $\mathbf{x}^{(\nu)} - \mathbf{x}^{(\nu-1)}$ . The parameters  $\alpha_\nu$  and  $\gamma_\nu$  are to be determined.

Using (9.3.5a), we rewrite (9.3.5b) in the form

$$\mathbf{p}^{(\nu)} = \mathbf{r}^{(\nu)} + \gamma_\nu \alpha_{\nu-1} \mathbf{p}^{(\nu-1)} = \mathbf{r}^{(\nu)} + \beta_{\nu-1} \mathbf{p}^{(\nu-1)} \quad (9.3.5c)$$

with the goal of specifying  $\alpha_\nu$  and  $\beta_{\nu-1}$  so that convergence is as fast as possible. As with steepest descent, we choose  $\alpha_\nu$  to minimize  $E(\mathbf{x}^{(\nu+1)})$ . In this case,

$$E(\mathbf{x}^{(\nu+1)}) = E(\mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{p}^{(\nu)}) = \frac{1}{2}(\mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{p}^{(\nu)})^T \mathbf{A}(\mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{p}^{(\nu)}) - \mathbf{b}^T(\mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{p}^{(\nu)}). \quad (9.3.6)$$

Differentiating with respect to  $\alpha_\nu$

$$E'(\mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{p}^{(\nu)}) = (\mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)} - \mathbf{b}^T \mathbf{p}^{(\nu)} = 0$$

or, using (9.3.2b)

$$(-\mathbf{r}^{(\nu)} + \alpha_\nu \mathbf{p}^{(\nu)} \mathbf{A})^T \mathbf{p}^{(\nu)} = 0.$$

Thus,

$$\alpha_\nu = \frac{(\mathbf{r}^{(\nu)})^T \mathbf{p}^{(\nu)}}{(\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}}, \quad \nu = 0, 1, \dots \quad (9.3.7)$$

Let's develop a few properties and relationships that will be interesting in their own right and useful for calculating  $\beta_\nu$ . First, we'll use (9.3.2b) and (9.3.5a) to write

$$\mathbf{r}^{(\nu+1)} = \mathbf{b} - \mathbf{A} \mathbf{x}^{(\nu+1)} = \mathbf{b} - \mathbf{A}(\mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{p}^{(\nu)}) = \mathbf{r}^{(\nu)} - \alpha_\nu \mathbf{A} \mathbf{p}^{(\nu)}. \quad (9.3.8)$$

Taking an inner product with  $\mathbf{p}^{(\nu)}$  to obtain

$$(\mathbf{r}^{(\nu+1)})^T \mathbf{p}^{(\nu)} = (\mathbf{r}^{(\nu)})^T \mathbf{p}^{(\nu)} - \alpha_\nu (\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}.$$

Using (9.3.7) to eliminate  $\alpha_\nu$  reveals the orthogonality condition

$$(\mathbf{r}^{(\nu+1)})^T \mathbf{p}^{(\nu)} = 0. \quad (9.3.9a)$$

Next, take the inner product of (9.3.5c) with  $\mathbf{r}^{(\nu+1)}$  to obtain

$$(\mathbf{r}^{(\nu+1)})^T \mathbf{p}^{(\nu+1)} = (\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu+1)} + \beta_\nu (\mathbf{r}^{(\nu+1)})^T \mathbf{p}^{(\nu)}$$

or, using (9.3.9a),

$$(\mathbf{r}^{(\nu+1)})^T \mathbf{p}^{(\nu+1)} = (\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu+1)}.$$

If we select  $\mathbf{p}^{(0)} = \mathbf{r}^{(0)}$ , then we may write the above expression as

$$(\mathbf{r}^{(\nu)})^T \mathbf{p}^{(\nu)} = (\mathbf{r}^{(\nu)})^T \mathbf{r}^{(\nu)}, \quad \nu = 0, 1, \dots \quad (9.3.9b)$$

Let us next expand (9.3.6) while using (9.3.1) and (9.3.2b) to show that

$$E(\mathbf{x}^{(\nu+1)}) = E(\mathbf{x}^{(\nu)}) - \alpha_\nu (\mathbf{r}^{(\nu)})^T \mathbf{p}^{(\nu)} + (1/2) \alpha_\nu^2 (\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}.$$

Using (9.3.7) and (9.3.9b)

$$E(\mathbf{x}^{(\nu+1)}) = E(\mathbf{x}^{(\nu)}) - \frac{1}{2} \frac{[(\mathbf{r}^{(\nu)})^T \mathbf{p}^{(\nu)}]^2}{(\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}} = E(\mathbf{x}^{(\nu)}) - \frac{1}{2} \frac{[(\mathbf{r}^{(\nu)})^T \mathbf{r}^{(\nu)}]^2}{(\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}}. \quad (9.3.10)$$

Having (9.3.10), we see that the error  $E(\mathbf{x}^{(\nu+1)})$  is decreased most rapidly when  $(\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}$  is minimal. This will be our criterion for determining  $\beta_\nu$ . Using (9.3.5c), we have

$$(\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)} = (\mathbf{r}^{(\nu)} + \beta_{\nu-1} \mathbf{p}^{(\nu-1)})^T \mathbf{A} (\mathbf{r}^{(\nu)} + \beta_{\nu-1} \mathbf{p}^{(\nu-1)}).$$

Minimizing with respect to  $\beta_{\nu-1}$  gives

$$\beta_{\nu-1} = - \frac{(\mathbf{r}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu-1)}}{(\mathbf{p}^{(\nu-1)})^T \mathbf{A} \mathbf{p}^{(\nu-1)}}$$

or reindexing

$$\beta_\nu = - \frac{(\mathbf{r}^{(\nu+1)})^T \mathbf{A} \mathbf{p}^{(\nu)}}{(\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}}, \quad \nu = 0, 1, \dots \quad (9.3.11)$$

Using (9.3.5c)

$$(\mathbf{p}^{(\nu+1)})^T \mathbf{A} \mathbf{p}^{(\nu)} = (\mathbf{r}^{(\nu+1)})^T \mathbf{A} \mathbf{p}^{(\nu)} + \beta_\nu (\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)},$$

which, upon use of (9.3.11), reveals

$$(\mathbf{p}^{(\nu+1)})^T \mathbf{A} \mathbf{p}^{(\nu)} = 0, \quad \nu = 0, 1, \dots \quad (9.3.12a)$$

Thus, the search directions are orthogonal with respect to a *strain energy inner product*. We usually call this a conjugacy condition and say that the search directions are *conjugate*.

Using (9.3.12a) with (9.3.5c), we have

$$(\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)} = (\mathbf{r}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)} + \beta_{\nu-1} (\mathbf{p}^{(\nu-1)})^T \mathbf{A} \mathbf{p}^{(\nu)} = (\mathbf{r}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}.$$

Combining this result with (9.3.8) yields

$$(\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu)} = (\mathbf{r}^{(\nu)})^T \mathbf{r}^{(\nu)} - \alpha_\nu (\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{r}^{(\nu)} = (\mathbf{r}^{(\nu)})^T \mathbf{r}^{(\nu)} - \alpha_\nu (\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}.$$

Using (9.3.7) and (9.3.9b), we find the orthogonality relation

$$(\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu)} = 0. \quad (9.3.12b)$$

Equation (9.3.11) can be put in a slightly simpler form by using (9.3.8) and (9.3.12b) to obtain

$$(\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu+1)} = (\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu)} - \alpha_\nu (\mathbf{r}^{(\nu+1)})^T \mathbf{A} \mathbf{p}^{(\nu)} = -\alpha_\nu (\mathbf{r}^{(\nu+1)})^T \mathbf{A} \mathbf{p}^{(\nu)}.$$

Using this with (9.3.11) and (9.3.7)

$$\beta_\nu = \frac{1}{\alpha_\nu} \frac{(\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu+1)}}{(\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}} = \frac{(\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu+1)}}{(\mathbf{p}^{(\nu)})^T \mathbf{r}^{(\nu)}}.$$

Finally, using (9.3.9b)

$$\beta_\nu = \frac{(\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu+1)}}{(\mathbf{r}^{(\nu)})^T \mathbf{r}^{(\nu)}}. \quad (9.3.12c)$$

We summarize our findings as a theorem.

**Theorem 9.3.1.** *The residuals and search directions of the conjugate gradient method satisfy*

$$(\mathbf{r}^{(\mu)})^T \mathbf{r}^{(\nu)} = (\mathbf{p}^{(\mu)})^T \mathbf{A} \mathbf{p}^{(\nu)} = \mathbf{0}, \quad \mu \neq \nu. \quad (9.3.13)$$

*Proof.* This has essentially been proven by the prior developments.  $\square$

An algorithm for the conjugate gradient method is presented in Figure 9.3.3. Some comments on the algorithm follow:

1. Equation (9.3.9b) was used to modify the expression (9.3.7) for  $\alpha_\nu$ .
2. The procedure requires storage for the nonzero elements of  $\mathbf{A}$  and for  $\mathbf{x}^{(\nu)}$ ,  $\mathbf{p}^{(\nu)}$ , and  $\mathbf{r}^{(\nu)}$ . An additional vector is needed to store the product  $\mathbf{A} \mathbf{p}^{(\nu)}$ . Thus, storage costs remain modest relative to the direct methods of Section 9.1.
3. The procedure requires a matrix multiplication ( $\mathbf{A} \mathbf{p}^{(\nu)}$ ) and computation of two inner products ( $(\mathbf{r}^{(\nu)})^T \mathbf{r}^{(\nu)}$  and  $(\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}$ ) per step.
4. Unlike SOR methods, there are no acceleration parameters to determine.

```

procedure conjugate_gradient
   $\mathbf{p}^{(0)} = \mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ 
   $\nu = 0$ 
  while not converged do
     $\alpha_\nu = (\mathbf{r}^{(\nu)})^T \mathbf{r}^{(\nu)} / (\mathbf{p}^{(\nu)})^T \mathbf{A} \mathbf{p}^{(\nu)}$ 
     $\mathbf{x}^{(\nu+1)} = \mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{p}^{(\nu)}$ 
     $\mathbf{r}^{(\nu+1)} = \mathbf{r}^{(\nu)} - \alpha_\nu \mathbf{A} \mathbf{p}^{(\nu)}$ 
     $\beta_\nu = (\mathbf{r}^{(\nu+1)})^T \mathbf{r}^{(\nu+1)} / (\mathbf{r}^{(\nu)})^T \mathbf{r}^{(\nu)}$ 
     $\mathbf{p}^{(\nu+1)} = \mathbf{r}^{(\nu+1)} + \beta_\nu \mathbf{p}^{(\nu)}$ 
     $\nu = \nu + 1$ 
  end while

```

Figure 9.3.3: Conjugate gradient algorithm.

The conjugate gradient method is both a direct and an iterative method as indicated by the following theorem.

**Theorem 9.3.2.** *Let  $\mathbf{A}$  be a positive definite, symmetric,  $N \times N$  matrix. Then the conjugate gradient method converges to the exact solution in no more than  $N$  steps.*

*Proof.* By Theorem 9.3.1, the residuals  $\mathbf{r}^{(\nu)}$ ,  $\nu = 0, 1, \dots, N-1$ , are mutually orthogonal. Since the space is  $N$ -dimensional, the residual  $\mathbf{r}^{(N)}$  must be zero; hence, the method converges in  $N$  steps.  $\square$

While convergence is achieved in  $N$  steps, the hope is to produce acceptable approximations of  $\mathbf{x}$  in far fewer than  $N$  steps when  $N$  is large. Practically, convergence may not be achieved in  $N$  steps when round-off errors are present.

*Example 9.3.2* ([5], Section 14.2). Consider the solution of Laplace's equation on a square region with  $\Delta x = \Delta y = h$ . Let the Dirichlet boundary conditions be prescribed so that the exact solution is

$$u(x, y) = e^x \sin y.$$

Solutions were calculated using SOR and conjugate gradient iterations until the change in the solution in the  $\mathcal{L}^2$  norm was less than  $10^{-7}$ . The value of  $\omega = 2/(1 + \pi h)$  was used with the SOR method. Results are shown in Table 9.3.1. The two methods are comparable. Apparent convergence is at a linear rate in  $h$ , *i.e.*, doubling  $h$  approximately

doubles the number of SOR and conjugate gradient iterations to convergence. Since an SOR iteration is less costly than a conjugate gradient step, we may infer that the SOR procedure is the faster.

$1/h$	SOR	CG
10	31	27
20	64	54
40	122	107

Table 9.3.1: Number of SOR and conjugate gradient iterations to convergence for Example 9.3.2.

The following theorem confirms the findings of the previous example.

**Theorem 9.3.3.** *Let  $\mathbf{A}$  be a symmetric and positive definite matrix, then iterates of the conjugate gradient method satisfy*

$$\|\mathbf{x}^{(\nu)} - \mathbf{x}\|_{\mathbf{A}} \leq 2 \left( \frac{\sqrt{\kappa_2} - 1}{\sqrt{\kappa_2} + 1} \right)^{\nu} \|\mathbf{x}^{(0)} - \mathbf{x}\|_{\mathbf{A}} \quad (9.3.14)$$

where  $\|\cdot\|_{\mathbf{A}}$  and  $\kappa_2$  were defined in (9.3.4a, 9.3.4c).

*Proof.* cf. [4], Section 6.11. □

Examining (9.3.14) and (9.3.4c), we see that convergence is fastest when the eigenvalues of  $\mathbf{A}$  are clustered together, *i.e.*, when  $\kappa_2(\mathbf{A}) \approx 1$ .

*Example 9.3.3.* The factor

$$R = \frac{\sqrt{\kappa_2} - 1}{\sqrt{\kappa_2} + 1}$$

determines the convergence rate of the conjugate gradient method. Since the condition number  $\kappa_2$  depends on the eigenvalues of  $\mathbf{A}$ , it would seem that we have to examine the eigenvalue problem

$$\mathbf{A}\mathbf{q} = \lambda\mathbf{q}.$$

Using (9.2.8a), let us write this relation as

$$(\mathbf{D} - \mathbf{L} - \mathbf{U})\mathbf{q} = \lambda\mathbf{q}$$

where  $\mathbf{D}$ ,  $\mathbf{L}$ , and  $\mathbf{U}$  were defined by (9.2.8b-9.2.8c). Multiplying by  $\mathbf{D}^{-1}$  and using (9.2.10b), we have

$$\mathbf{M}_J \mathbf{q} = (\mathbf{I} - \lambda \mathbf{D}^{-1}) \mathbf{q}$$

where  $\mathbf{M}_J$  is the Jacobi iteration matrix. For the Laplacian operator,  $\mathbf{D} = \mathbf{I}$  and we have  $\mu = 1 - \lambda$ , where  $\mu$  is an eigenvalue of  $\mathbf{M}_J$ . Still confining our attention to the Laplacian operator, we may use (9.2.17d) to evaluate  $\mu$  and, hence, obtain

$$\lambda = 1 - \mu = 4\theta_x \sin^2 \frac{m\pi}{2J} + 4\theta_y \sin^2 \frac{n\pi}{2K}, \quad m = 1, 2, \dots, J-1, \quad n = 1, 2, \dots, K-1.$$

For simplicity, let us focus on a square grid ( $J = K$ ) where

$$\lambda = \sin^2 \frac{m\pi}{2J} + \sin^2 \frac{n\pi}{2J}, \quad m, n = 1, 2, \dots, J-1.$$

The smallest eigenvalue occurs with  $m = n = 1$  and the largest occurs with  $m = n = J-1$ ; thus,

$$\lambda_{min} = 2 \sin^2 \frac{\pi}{2J}, \quad \lambda_{max} = 2 \sin^2 \frac{(J-1)\pi}{2J}.$$

Hence, using (9.3.4c)

$$\kappa_2 = \frac{\sin^2(J-1)\pi/2J}{\sin^2 \pi/2J}.$$

When  $J \gg 1$  we may approximate this as

$$\sqrt{\kappa_2} = \frac{\sin(J-1)\pi/2J}{\sin \pi/2J} \approx \frac{2J}{\pi}.$$

Thus,

$$R \approx \frac{1 - \pi/2J}{1 + \pi/2J} \approx 1 - \frac{\pi}{J}.$$

Convergence is, therefore, at the same rate as the SOR method (Example 9.2.9).

### 9.3.2 Preconditioned Conjugate Gradient Iterations

From Theorem 9.3.3, we see that the performance of the conjugate gradient method improves when the eigenvalues of  $\mathbf{A}$  are clustered about a point. This suggests the possibility of preconditioning  $\mathbf{A}$  by a positive definite matrix  $\mathbf{M}$  and solving

$$\mathbf{M}^{-1} \mathbf{A} \mathbf{x} = \mathbf{M}^{-1} \mathbf{b}. \quad (9.3.15)$$

If the eigenvalues of  $\mathbf{M}^{-1}\mathbf{A}$  were clustered, the conjugate gradient procedure may converge at a faster rate. The preconditioner  $\mathbf{M}$  should be chosen to minimize the solution time. There are, however, competing priorities. Thus, for example, the optimal choice of  $\mathbf{M}$  as far as clustering eigenvalues is concerned is  $\mathbf{M} = \mathbf{A}$ . This choice requires a direct solution of the original system and, thus, has an extreme cost. The optimal choice of  $\mathbf{M}$  as far as computational effort is concerned is  $\mathbf{M} = \mathbf{I}$ . This is the conjugate gradient algorithm and, thus, no improvement has been provided. The search for the best preconditioning is still an active area of research with optimality dependent on many factors including sparsity and intended computer architecture.

The preconditioning shown in (9.3.15) is called a *left preconditioning*. A *right preconditioning* is

$$\mathbf{A}\mathbf{M}^{-1}\mathbf{w} = \mathbf{b}, \quad \mathbf{x} = \mathbf{M}^{-1}\mathbf{w}. \quad (9.3.16a)$$

A *symmetric preconditioning* is

$$\mathbf{C}^{-1}\mathbf{A}\mathbf{C}^{-T}\mathbf{w} = \mathbf{C}^{-1}\mathbf{b}, \quad \mathbf{x} = \mathbf{C}^{-T}\mathbf{w} \quad (9.3.16b)$$

where  $\mathbf{C}^{-T}$  denotes the transpose of  $\mathbf{C}^{-1}$ . The preconditioning matrix  $\mathbf{C}$  need not be symmetric since  $\mathbf{C}^{-1}\mathbf{A}\mathbf{C}^{-T}$  is symmetric and positive definite when  $\mathbf{A}$  is. The matrix  $\mathbf{C}$  may be a Cholesky factor of  $\mathbf{M}$ , *i.e.*,

$$\mathbf{M} = \mathbf{C}\mathbf{C}^T. \quad (9.3.16c)$$

In this case,  $\mathbf{C}$  would be lower triangular and it could be obtained from the symmetric factorization of  $\mathbf{M}$  given by (9.1.8, 9.1.9) or (9.1.12, 9.3.11).

The preconditioned conjugate gradient (PCG) algorithm with the symmetric preconditioning may be implemented by applying the conjugate gradient procedure to

$$\tilde{\mathbf{A}}\mathbf{w} = \tilde{\mathbf{b}} \quad (9.3.17a)$$

where

$$\tilde{\mathbf{A}} = \mathbf{C}^{-1}\mathbf{A}\mathbf{C}^{-T}, \quad \tilde{\mathbf{b}} = \mathbf{C}^{-1}\mathbf{b}, \quad \mathbf{w} = \mathbf{C}^T\mathbf{x} \quad (9.3.17b)$$

```

ν = 0
while not converged do
  αν = (r̃(ν))Tr̃(ν) / (p̃(ν))TÃp̃(ν)
  w(ν+1) = w(ν) + ανp̃(ν)
  r̃(ν+1) = r̃(ν) - ανÃp̃(ν)
  βν = (r̃(ν+1))Tr̃(ν+1) / (r̃(ν))Tr̃(ν)
  p̃(ν+1) = r̃(ν+1) + βνp̃(ν)
  ν = ν + 1
end while

```

Figure 9.3.4: Main loop of the conjugate gradient algorithm applied to (9.3.17).

The main loop of the conjugate gradient algorithm of Figure 9.3.3 is reproduced in Figure 9.3.4 for the system (9.3.17). In this algorithm

$$\tilde{\mathbf{r}}^{(\nu)} = \tilde{\mathbf{b}} - \tilde{\mathbf{A}}\mathbf{w}^{(\nu)} = \mathbf{C}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}^{(\nu)}) = \mathbf{C}^{-1}\mathbf{r}^{(\nu)} \quad (9.3.18a)$$

Also

$$\tilde{\mathbf{p}}^{(\nu)} = \mathbf{C}^T\mathbf{p}^{(\nu)}. \quad (9.3.18b)$$

Using (9.3.18), let us rewrite the conjugate gradient algorithm of Figure 9.3.4 in terms of the original variables as

$$(\tilde{\mathbf{r}}^{(\nu)})^T\tilde{\mathbf{r}}^{(\nu)} = (\mathbf{r}^{(\nu)})^T\mathbf{C}^{-T}\mathbf{C}^{-1}\mathbf{r}^{(\nu)} = (\mathbf{r}^{(\nu)})^T\mathbf{M}^{-1}\mathbf{r}^{(\nu)}, \quad (9.3.19a)$$

$$\tilde{\mathbf{A}}\tilde{\mathbf{p}}^{(\nu)} = \mathbf{C}^{-1}\mathbf{A}\mathbf{C}^{-T}\mathbf{C}^T\mathbf{p}^{(\nu)} = \mathbf{C}^{-1}\mathbf{A}\mathbf{p}^{(\nu)} \quad (9.3.19b)$$

$$(\tilde{\mathbf{p}}^{(\nu)})^T\tilde{\mathbf{A}}\tilde{\mathbf{p}}^{(\nu)} = (\mathbf{p}^{(\nu)})^T\mathbf{C}\mathbf{C}^{-1}\mathbf{A}\mathbf{p}^{(\nu)} = (\mathbf{p}^{(\nu)})^T\mathbf{A}\mathbf{p}^{(\nu)} \quad (9.3.19c)$$

The PCG algorithm written in terms of the original variables appears in Figure 9.3.5. Some comments follow:

1. The PCG algorithm does not involve  $\mathbf{C}$  or a Cholesky factorization of  $\mathbf{M}$ . Thus, a PCG algorithm with the left preconditioning (9.3.15) would be identical. The PCG algorithm with the right preconditioning also gives the same sequence of operations ([4], Section 9.2).

```

procedure pcg
   $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$ 
  Solve  $\mathbf{M}\mathbf{z}^{(0)} = \mathbf{r}^{(0)}$ 
   $\mathbf{p}^{(0)} = \mathbf{z}^{(0)}$ 
   $\nu = 0$ 
  while not converged do
     $\alpha_\nu = (\mathbf{r}^{(\nu)})^T \mathbf{z}^{(\nu)} / (\mathbf{p}^{(\nu)})^T \mathbf{A}\mathbf{p}^{(\nu)}$ 
     $\mathbf{x}^{(\nu+1)} = \mathbf{x}^{(\nu)} + \alpha_\nu \mathbf{p}^{(\nu)}$ 
     $\mathbf{r}^{(\nu+1)} = \mathbf{r}^{(\nu)} - \alpha_\nu \mathbf{A}\mathbf{p}^{(\nu)}$ 
    Solve  $\mathbf{M}\mathbf{z}^{(\nu+1)} = \mathbf{r}^{(\nu+1)}$ 
     $\beta_\nu = (\mathbf{r}^{(\nu+1)})^T \mathbf{z}^{(\nu+1)} / (\mathbf{r}^{(\nu)})^T \mathbf{z}^{(\nu)}$ 
     $\mathbf{p}^{(\nu+1)} = \mathbf{z}^{(\nu+1)} + \beta_\nu \mathbf{p}^{(\nu)}$ 
     $\nu = \nu + 1$ 
  end while

```

Figure 9.3.5: PCG algorithm.

2. Rather than compute  $\mathbf{M}^{-1}$ , it is more efficient to solve  $\mathbf{M}\mathbf{z}^{(\nu)} = \mathbf{r}^{(\nu)}$  for  $\mathbf{z}^{(\nu)}$ .
3. The original conjugate gradient algorithm of Figure 9.3.3 would set  $\tilde{\mathbf{p}}^{(0)} = \tilde{\mathbf{r}}^{(0)}$ . Using (9.3.18a, 9.3.18b)

$$\mathbf{C}^T \mathbf{p}^{(0)} = \mathbf{C}^{-1} \mathbf{r}^{(0)}$$

or, using (9.3.16c)

$$\mathbf{p}^{(0)} = \mathbf{C}^{-T} \mathbf{C}^{-1} \mathbf{r}^{(0)} = \mathbf{M}^{-1} \mathbf{r}^{(0)} = \mathbf{z}^{(0)}.$$

This explains the choice of  $\mathbf{p}^{(0)}$  in the initial stages of the algorithm. The subsequent calculation of  $\mathbf{p}^{(\nu+1)}$  uses the same manipulations.

4. Storage is needed for  $\mathbf{A}$ ,  $\mathbf{M}$  (possibly in factored form),  $\mathbf{x}^{(\nu)}$ ,  $\mathbf{p}^{(\nu)}$ ,  $\mathbf{r}^{(\nu)}$ ,  $\mathbf{z}^{(\nu)}$ , and  $\mathbf{A}\mathbf{p}^{(\nu)}$ . Relative to the conjugate gradient procedure, additional storage is needed for  $\mathbf{M}$  and  $\mathbf{z}^{(\nu)}$ . Storage costs are still modest relative to those of direct methods.
5. In addition to the matrix multiplication ( $\mathbf{A}\mathbf{p}^{(\nu)}$ ) and the two inner products per step required by the conjugate gradient method, a linear equations solution ( $\mathbf{M}\mathbf{z}^{(\nu+1)} = \mathbf{r}^{(\nu+1)}$ ) is required

6. Using (9.3.13) for the conjugate gradient method

$$(\tilde{\mathbf{r}}^{(\mu)})^T \tilde{\mathbf{r}}^{(\nu)} = 0, \quad (\tilde{\mathbf{p}}^{(\mu)})^T \tilde{\mathbf{A}} \tilde{\mathbf{p}}^{(\nu)} = 0, \quad \nu \neq \mu.$$

Using (9.3.18a, 9.3.18b), we obtain the “orthogonality conditions”

$$(\mathbf{r}^{(\mu)})^T \mathbf{M}^{-1} \mathbf{r}^{(\nu)} = 0, \quad (\mathbf{p}^{(\mu)})^T \mathbf{A} \mathbf{p}^{(\nu)} = 0, \quad \nu \neq \mu \quad (9.3.20)$$

7. When  $\mathbf{M}$  is positive definite

$$(\mathbf{r}^{(\nu)})^T \mathbf{z}^{(\nu)} = (\mathbf{r}^{(\nu)})^T \mathbf{M}^{-1} \mathbf{r}^{(\nu)} > 0.$$

Thus, values of  $\beta_\nu$  can always be obtained and the procedure does not fail.

Let us select some preconditionings, beginning with some choices based on iterative strategies. It will be convenient to write the basic fixed-point strategy (9.2.2) in the form

$$\hat{\mathbf{M}} \mathbf{x}^{(\nu+1)} = \hat{\mathbf{N}} \mathbf{x}^{(\nu)} + \mathbf{b} \quad (9.3.21a)$$

where

$$\mathbf{A} = \hat{\mathbf{M}} - \hat{\mathbf{N}} \quad (9.3.21b)$$

Comparing (9.3.21) with (9.2.10b), (9.2.15b), and (9.2.21d), we have

- *Jacobi iteration:*

$$\hat{\mathbf{M}}_J = \mathbf{D}, \quad \hat{\mathbf{N}}_J = \mathbf{L} + \mathbf{U} \quad (9.3.22)$$

- *Gauss-Seidel iteration:*

$$\hat{\mathbf{M}}_{GS} = \mathbf{D} - \mathbf{L}, \quad \hat{\mathbf{N}}_{GS} = \mathbf{U} \quad (9.3.23)$$

- *SOR Iteration:*

$$\hat{\mathbf{M}}_\omega = \frac{1}{\omega}(\mathbf{D} - \omega \mathbf{L}), \quad \hat{\mathbf{N}}_\omega = \frac{1 - \omega}{\omega} \mathbf{D} + \mathbf{U}. \quad (9.3.24)$$

Recall that  $\mathbf{D}$  is the diagonal part,  $\mathbf{L}$  is the negative of the lower triangular part, and  $\mathbf{U}$  is the negative of the upper triangular part of  $\mathbf{A}$  (*cf.* (9.2.8)). Let us also include symmetric successive over relaxation (SSOR) in our study. As discussed in Section 9.2, SSOR takes two SOR sweeps with the unknowns placed in reverse order on the second sweep. Using (9.3.24) and (9.3.21), the first step of the SOR procedure is

$$(\mathbf{D} - \omega\mathbf{L})\mathbf{x}^{(\nu+1/2)} = [(1 - \omega)\mathbf{D} + \omega\mathbf{U}]\mathbf{x}^{(\nu)} + \omega\mathbf{b}. \quad (9.3.25a)$$

Reversing the sweep direction on the second step yields

$$(\mathbf{D} - \omega\mathbf{U})\mathbf{x}^{(\nu+1)} = [(1 - \omega)\mathbf{D} + \omega\mathbf{L}]\mathbf{x}^{(\nu+1/2)} + \omega\mathbf{b}. \quad (9.3.25b)$$

The intermediate solution  $\mathbf{x}^{(\nu+1/2)}$  can be eliminated to obtain a scheme of the form (9.3.21) with (*cf.* Problem 2 at the end of this section)

$$\hat{\mathbf{M}}_{SSOR} = \frac{1}{\omega(2 - \omega)}(\mathbf{D} - \omega\mathbf{L})\mathbf{D}^{-1}(\mathbf{D} - \omega\mathbf{U}), \quad (9.3.26a)$$

$$\hat{\mathbf{N}}_{SSOR} = \frac{1}{\omega}[\mathbf{I} - \frac{1}{2 - \omega}(\mathbf{D} - \omega\mathbf{L})\mathbf{D}^{-1}][\omega\mathbf{U} + (1 - \omega)\mathbf{D}]. \quad (9.3.26b)$$

At the moment, the iteration matrix  $\hat{\mathbf{M}}$  of (9.3.21) and the preconditioning matrix  $\mathbf{M}$  of (9.3.15) are unrelated; however, observe that the exact solution of (9.3.21a) satisfies

$$\hat{\mathbf{M}}\mathbf{x} = \hat{\mathbf{N}}\mathbf{x} + \mathbf{b}.$$

Multiplying by  $\hat{\mathbf{M}}^{-1}$

$$(\mathbf{I} - \hat{\mathbf{M}}^{-1}\hat{\mathbf{N}})\mathbf{x} = \hat{\mathbf{M}}^{-1}\mathbf{b}.$$

Using (9.3.21b) to eliminate  $\hat{\mathbf{N}}$

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

This has the same form as the left preconditioning (9.3.15); thus,  $\hat{\mathbf{M}}$  serves as a preconditioner.

Examining (9.3.22 - 9.3.24), (9.3.26), however, we see that  $\hat{\mathbf{M}}_{GS}$  and  $\hat{\mathbf{M}}_{\omega}$  are not symmetric. Thus, only the Jacobi and SSOR methods will furnish acceptable preconditionings and, of the two, we focus on the SSOR preconditioner (9.3.26a).

At each PCG iteration (Figure 9.3.5) we must solve

$$\hat{\mathbf{M}}_{SSOR} \mathbf{z}^{(\nu)} = \mathbf{r}^{(\nu)}. \quad (9.3.27a)$$

If  $\mathbf{A}$  is symmetric then  $\mathbf{U} = \mathbf{L}^T$  and (9.3.26a) becomes

$$\hat{\mathbf{M}}_{SSOR} = \frac{1}{\omega(2-\omega)} (\mathbf{D} - \omega \mathbf{L}) \mathbf{D}^{-1} (\mathbf{D} - \omega \mathbf{L}^T), \quad (9.3.27b)$$

Thus, (9.3.27a) may be solved with a forward, diagonal, and backward substitution as

$$(\mathbf{D} - \omega \mathbf{L}) \bar{\mathbf{z}} = \omega(2-\omega) \mathbf{r}^{(\nu)}, \quad (9.3.27c)$$

$$\mathbf{D}^{-1} \tilde{\mathbf{z}} = \bar{\mathbf{z}}, \quad (9.3.27d)$$

and

$$(\mathbf{D} - \omega \mathbf{L}^T) \mathbf{z}^{(\nu)} = \tilde{\mathbf{z}}. \quad (9.3.27e)$$

The choice of  $\omega$  doesn't appear to be critical and may, *e.g.*, be selected as unity.

*Example 9.3.4* ([5], Section 14.5). Consider the solution of Poisson's equation on a square with uniform spacing  $\Delta x = \Delta y = h$ . Suppose that the forcing and boundary data is such that the exact solution is

$$u(x, y) = \cos x \sin y.$$

The initial iterate was trivial inside the square and all other numerical parameters were selected as for Example 9.3.2. Comparisons of results obtained using SOR, CG, and PCG are presented in Table 9.3.2. The number of iterations of the SOR and CG algorithms is increasing as  $1/h$  while that of the PCG algorithm is increasing as  $1/h^{1/2}$ . The work of the conjugate gradient method is about twice that of the SOR method and that of the PCG method is about four times the SOR method. Thus, for small systems, the SOR and conjugate gradient method will be superior, but the SSOR-PCG method overtakes these methods for larger systems.

One of the most successful preconditioning techniques utilizes incomplete factorization by Gaussian elimination. Thus, let

$$\mathbf{M} = \mathbf{L}_0 \mathbf{D}_0 \mathbf{L}_0^T \quad (9.3.28)$$

$1/h$	SOR	CG	PCG
10	33	26	12
20	60	52	16
40	115	103	22

Table 9.3.2: Number of SOR, conjugate gradient, and SSOR-PCG iterations for Example 9.3.4.

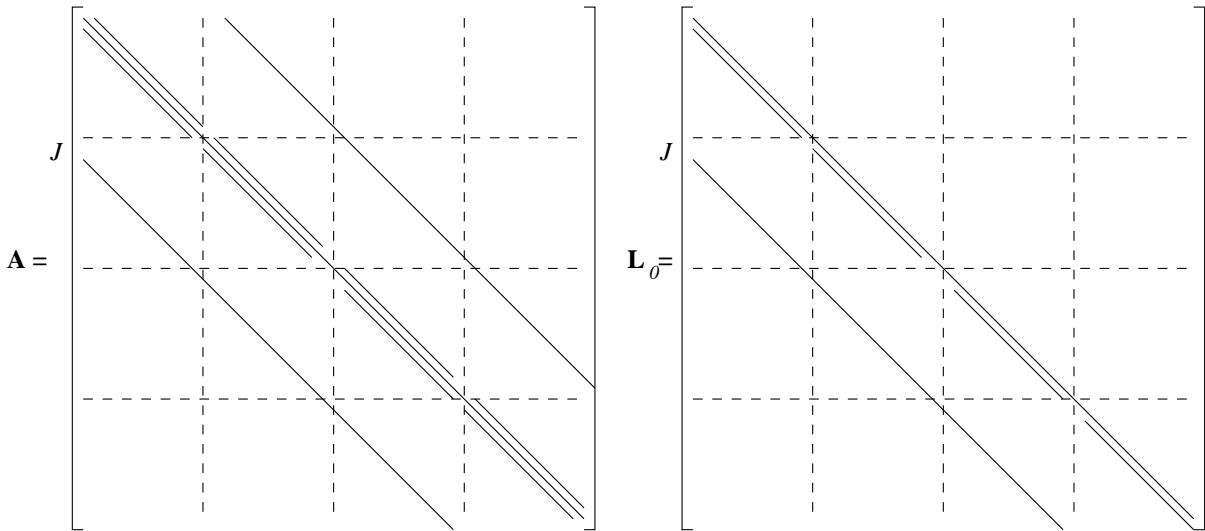


Figure 9.3.6: Nonzero structure of a discrete Poisson operator  $\mathbf{A}$  (left) and an approximate lower triangular factor  $\mathbf{L}_0$  (right).

where  $\mathbf{L}_0$  is determined to have a particular sparsity structure.

*Example 9.3.5.* Consider a Dirichlet problem for Poisson's equation, which has the nonzero structure shown in Figure 9.3.6. Ordering the equations by rows, we have

$$(\mathbf{A}\mathbf{x})_n = -x_n + \theta_x(x_{n-1} + x_{n+1}) + \theta_y(x_{n-J} + x_{n+J}). \quad (9.3.29a)$$

If equation  $n$  corresponds to mesh point  $(j, k)$ , we may also write (9.3.29a) in the (double-subscripted) mesh notation as

$$(\mathbf{A}\mathbf{x})_{jk} = -x_{jk} + \theta_x(x_{j-1,k} + x_{j+1,k}) + \theta_y(x_{j,k-1} + x_{j,k+1}). \quad (9.3.29b)$$

We insist that

$$(\mathbf{L}_0\mathbf{x})_n = x_n + b_n x_{n-1} + c_n x_{n-J}. \quad (9.3.30a)$$

Taking a transpose

$$(\mathbf{L}_0^T\mathbf{x})_n = x_n + b_{n+1} x_{n+1} + c_{n+J} x_{n+J}. \quad (9.3.30b)$$

Likewise

$$(\mathbf{D}_0 \mathbf{x})_n = d_n x_n. \quad (9.3.30c)$$

Multiplying (9.3.30b) and (9.3.30c)

$$(\mathbf{D}_0 \mathbf{L}_0^T \mathbf{x})_n = d_n (x_n + b_{n+1} x_{n+1} + c_{n+J} x_{n+J}).$$

Now, using (9.3.30a)

$$(\mathbf{L}_0 \mathbf{D}_0 \mathbf{L}_0^T \mathbf{x})_n = (\mathbf{D}_0 \mathbf{L}_0^T \mathbf{x})_n + b_n (\mathbf{D}_0 \mathbf{L}_0^T \mathbf{x})_{n-1} + c_n (\mathbf{D}_0 \mathbf{L}_0^T \mathbf{x})_{n-J}.$$

Combining the above two equations

$$\begin{aligned} (\mathbf{L}_0 \mathbf{D}_0 \mathbf{L}_0^T \mathbf{x})_n &= d_n (x_n + b_{n+1} x_{n+1} + c_{n+J} x_{n+J}) + b_n d_{n-1} (x_{n-1} + b_n x_n + c_{n+J-1} x_{n+J-1}) + \\ &\quad c_n d_{n-J} (x_{n-J} + b_{n+1-J} x_{n+1-J} + c_n x_n). \end{aligned}$$

Regrouping terms

$$\begin{aligned} (\mathbf{L}_0 \mathbf{D}_0 \mathbf{L}_0^T \mathbf{x})_n &= (d_n + d_{n-1} b_n^2 + d_{n-J} c_n^2) x_n + d_{n-1} b_n x_{n-1} + d_n b_{n+1} x_{n+1} + \\ &\quad d_n c_{n+J} x_{n+J} + d_{n-J} c_n x_{n-J} + b_{n+1-J} d_{n-J} c_n x_{n+1-J} + b_n d_{n-1} c_{n+J-1} x_{n+J-1}. \end{aligned} \quad (9.3.31)$$

Thus,  $\mathbf{L}_0 \mathbf{D}_0 \mathbf{L}_0^T$  has seven non-trivial bands as shown in Figure 9.3.7.

Using incomplete factorization, we equate the coefficients of  $x_n$ ,  $x_{n+1}$ , and  $x_{n+J}$  in (9.3.31) to those of  $(\mathbf{A} \mathbf{x})_n$  in (9.3.29a). This yields

$$d_n = -1 - d_{n-1} b_n^2 - d_{n-J} c_n^2, \quad n = 1, 2, \dots, N. \quad (9.3.32a)$$

$$b_{n+1} = \theta_x / d_n, \quad n = 1, 2, \dots, N-1. \quad (9.3.32b)$$

$$c_{n+J} = \theta_y / d_n, \quad n = 1, 2, \dots, N-J. \quad (9.3.32c)$$

*Example 9.3.6.* ([5], Section 14.5). Consider the solution of Laplace's equation on the unit square using a square mesh ( $\theta_x = \theta_y = 1/4$ ) with  $h \times h$  spacing by the 9-point approximation of the Laplacian

$$\frac{1}{6} (U_{j+1, k+1} + U_{j+1, k-1} + U_{j-1, k+1} + U_{j-1, k-1})$$

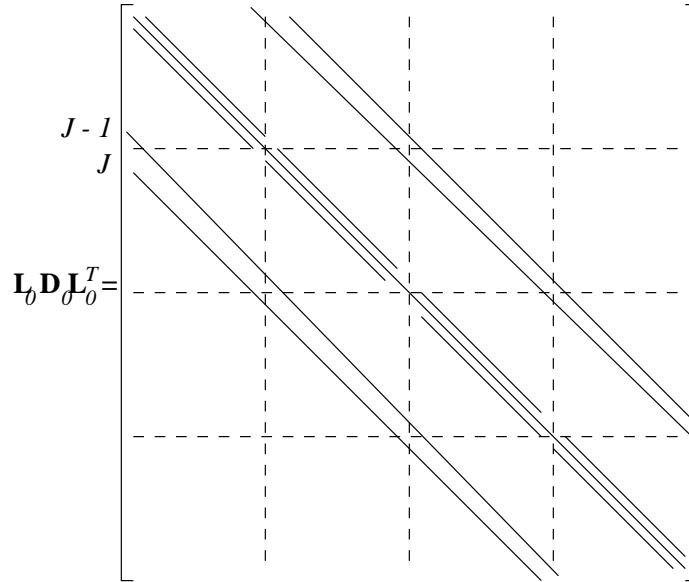


Figure 9.3.7: The seven-band structure of  $\mathbf{L}_0 \mathbf{D}_0 \mathbf{L}_0^T$ .

$$+\frac{2}{3}(U_{j+1,k} + U_{j-1,k} + U_{j,k+1} + U_{j,k-1}) - \frac{10}{3}U_{jk} = 0.$$

The 9-point approximation of the Laplacian is accurate to  $O(h^4)$  compared to the  $O(h^2)$  accuracy of the 5-point formula. The Dirichlet boundary conditions are established so that the exact solution is

$$u(x, y) = e^{3x} \sin 3y.$$

The problem is solved using the PCG method with the following preconditioners:

- *None*. No preconditioning,
- *SSOR-5*. SSOR iteration using the 5-point approximation of the Laplacian,
- *SSOR-9*. SSOR iteration using the 9-point approximation of the Laplacian, and
- *ILU(0)-5*. Incomplete Cholesky factorization of the 5-point Laplacian with no fill in.

Each method was started with a trivial solution in the interior and one that satisfied the exact solution on the boundary. The iteration was terminated when the difference between successive iterates in the  $\mathcal{L}^2$  norm was less than  $10^{-10}$ . Each SSOR method used  $\omega = 2/(1 + \pi h)$ . The number of iterations to reach convergence are recorded in Table 9.3.2.

$1/h$	None	ILU(0)-5	SSOR-5	SSOR-9
10	28	16	18	16
20	57	28	25	23
40	112	52	34	32

Table 9.3.3: Number of iterations to convergence for the PCG solution of Example 9.3.6 using the indicated preconditioners.

The number of iterations for the two SSOR preconditioners increases as  $O(h^{-1/2})$ . The more accurate SSOR-9 iterative preconditioning differs little from the SSOR-5 preconditioning. The number of iterations for the incomplete factorization preconditioner is increasing slower than  $O(1/h)$  but faster than  $O(h^{-1/2})$ . All preconditioners in the table are better than using the conjugate gradient method without preconditioning.

Thus far, we have used incomplete factorization without allowing any fill in. In Example 9.3.6, we referred to this preconditioning as ILU(0). The ILU(0) preconditioning may not provide enough acceleration to the basic conjugate gradient algorithm. Thus, we consider continuing the factorization to additional levels by allowing some fill in. We will call a preconditioner ILU( $p$ ) if there are  $p$  levels of fill in. The next preconditioner beyond ILU(0) allows one level of fill in. The structure of  $\mathbf{L}_1$  for this ILU(1) algorithm is shown on the right of Figure 9.3.8. One row nearest the bandwidth has been filled in. It's simplest to imagine  $\mathbf{A}$  as having the two bands corresponding to the filled band of  $\mathbf{L}_1$  as also being filled in. This situation is shown on the upper right of Figure 9.3.8. The product  $\mathbf{L}_1\mathbf{D}\mathbf{L}_1^T$  fills in yet another band (*cf.* the lower portion of Figure 9.3.8), which is ignored in the elimination.

An algorithm for performing incomplete factorization to level  $p$  for elliptic problems on rectangular meshes follows the procedure described in Example 9.3.5. Let's, instead, illustrate the method for a more general sparse matrix. This is typically done by defining a *level of fill* for each element created by Gaussian elimination. The assumption is that elements get smaller as the fill in progresses. A size  $\epsilon^k$ ,  $0 < \epsilon < 1$ , is assigned to an element at fill level  $k$ . Initially, a nonzero element has a unit level of fill and a zero element has an infinite level of fill. Rather than work with the  $\mathbf{LDL}^T$  factorization, let's simplify our task and work with a more general  $\mathbf{LU}$  factorization such as (9.1.3). The

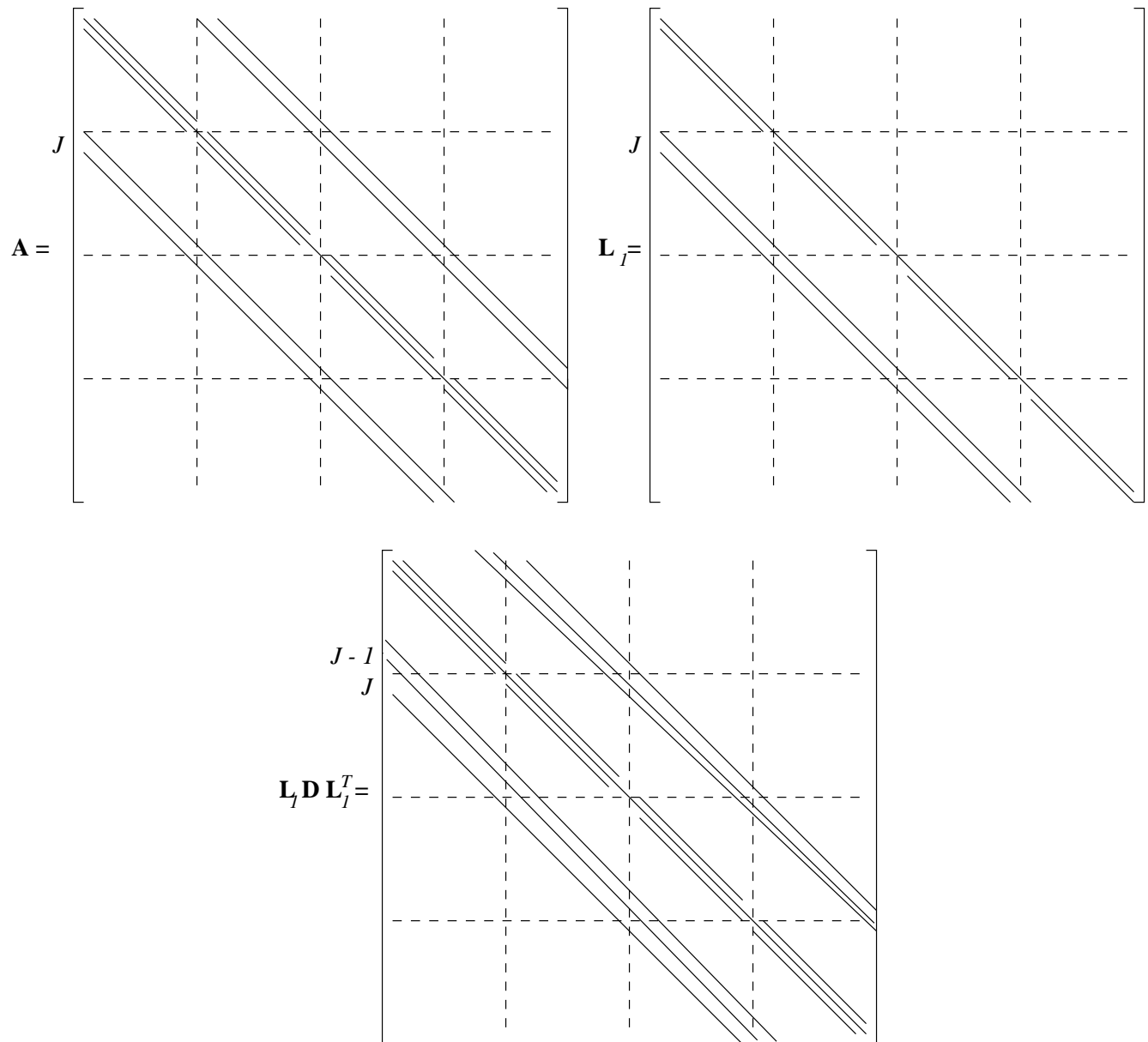


Figure 9.3.8: Imagined structure of  $\mathbf{A}$  (top left) and structure of  $\mathbf{L}_1$  (top right) for an ILU(1) preconditioner. The fill in when calculating  $\mathbf{L}_1 \mathbf{D} \mathbf{L}_1^T$  is shown at the bottom.

computation is generated within a loop and, typically, the elements of  $\mathbf{L}$  replace the elements in the lower triangular portion of  $\mathbf{A}$  and the elements of  $\mathbf{U}$  replace the elements in the upper triangular portion of  $\mathbf{A}$ . A sample loop structure for the elimination is shown in Figure 9.3.9. This is very basic Gaussian elimination. No pivoting is performed and sparsity is not included.

```

for  $i = 2$  to  $N$  do
  for  $k = 1$  to  $i - 1$  do
     $a_{ik} = a_{ik}/a_{kk}$ 
    for  $j = k + 1$  to  $N$  do
       $a_{ij} = a_{ij} - a_{ik}a_{kj}$ 
    end for
  end for
end for

```

Figure 9.3.9: Gaussian elimination loop.

The Gaussian elimination update in Figure 9.3.9 is

$$a_{ij} := a_{ij} - a_{ik}a_{kj}.$$

Let the current level of fill of  $a_{ij}$  be denoted as  $lev_{ij}$ . Then the size of updated element  $a_{ij}$  should be

$$\epsilon^{lev_{ij}} - \epsilon^{lev_{ik}}\epsilon^{lev_{kj}} = \epsilon^{lev_{ij}} - \epsilon^{(lev_{ik}+lev_{kj})}.$$

Roughly speaking, the size of the updated element  $a_{ij}$  will be the maximum of the two sizes  $\epsilon^{lev_{ij}}$  and  $\epsilon^{(lev_{ik}+lev_{kj})}$ . Thus, it makes sense to define the new level of fill as

$$lev_{ij} := \min(lev_{ij}, lev_{ik} + lev_{kj}).$$

It is common (*cf.* [4], Section 10.3) to shift all levels by -1 from this definition. Thus, the initial level of fill of  $a_{ij}$  is set as

$$lev_{ij} := \begin{cases} 0, & \text{if } a_{ij} \neq 0, \text{ or } i = j \\ \infty, & \text{otherwise} \end{cases}. \quad (9.3.33a)$$

The updated level of fill is

$$lev_{ij} := \min(lev_{ij}, lev_{ik} + lev_{kj} + 1). \quad (9.3.33b)$$

We see that  $lev_{ij}$  cannot increase during the factorization. Thus, if  $a_{ij} \neq 0$  in the original matrix  $\mathbf{A}$ ,  $lev_{ij}$  never becomes nonzero. Equations (9.3.33) suggest a natural way of discarding elements during the incomplete factorization. With an ILU( $p$ ) algorithm, we'll keep all elements whose level of fill does not exceed  $p$ . The *zero pattern* of  $\mathbf{A}$  for the ILU( $p$ ) procedure may be defined as the set

$$\mathcal{Z}_p(\mathbf{A}) := \{(i, j) | lev_{ij} > p\}. \quad (9.3.33c)$$

This definition is consistent with the ILU(0) preconditioning described in Example 9.3.5.

```

Define  $lev_{ij}$  according to (9.3.33a)
for  $i = 2$  to  $N$  do
  for  $k = 1$  to  $i - 1$  do
    if  $lev_{ik} \leq p$  then
       $a_{ik} = a_{ik}/a_{kk}$ 
      for  $j = k + 1$  to  $N$  do
        if  $lev_{ij} \leq p$  then
           $a_{ij} = a_{ij} - a_{ik}a_{kj}$ 
          Update  $lev_{ij}$  according to (9.3.33b)
        end if
      end for
    end for
  for  $j = k + 1$  to  $N$  do
    if  $lev_{ij} > p$  then
       $a_{ij} = 0;$ 
       $lev_{ij} = \infty$ 
    end if
  end for
end if
end for

```

Figure 9.3.10: Incomplete ILU( $p$ ) factorization.

A pseudocode ILU( $p$ ) algorithm appears in Figure 9.3.10. It is intended to give the general idea of the procedure and, as stated, is not very efficient. It ignores sparsity and does too much searching for levels. These conditions can be remedied. For example, the search for levels does not depend on the values of the elements of  $\mathbf{A}$  and can be separated from the elimination process. It can, thus, be done in advance and be used for several matrices having the same structure. However, because the algorithm is based on levels of fill rather than numerical information, some large elements of  $\mathbf{A}$  may be dropped because of their locations rather than their sizes. This may result in a poor preconditioning and, hence, require more iterations than necessary for convergence.

A *threshold elimination procedure* (ILUT) would identify small elements of  $\mathbf{A}$  and set them to zero during the factorization. At its simplest, this can be done by modifying the algorithm of Figure 9.3.10 as follows:

1. eliminate the definition and updates of  $lev_{ij}$ ,
2. replace the tests on  $lev_{ik} \leq p$  and  $lev_{ij} \leq p$  by tests on  $a_{ik} \neq 0$  and  $a_{ij} \neq 0$ , respectively,
3. insert a rule for dropping  $a_{ik}$  following its calculation ( $a_{ik} = a_{ik}/a_{kk}$ ), and
4. replace the statements within the second  $j$  loop by rules for dropping  $a_{ij}$ .

Rules for dropping elements are described by Saad [4], Section 10.4. Briefly, he suggests

1. dropping elements  $a_{ik}$  (Item 3 above) and  $a_{ij}$  (Item 4 above) that are less than a tolerance  $\tau$  relative to the size (*e.g.*, the  $\mathcal{L}^2$  norm) of row  $i$ , and
2. only retaining the  $p$  largest elements  $a_{ij}$  (Item 4 above) of row  $i$  without dropping the diagonal element.

The goal of the second dropping rule is to control the number of elements per row (or column) of  $\mathbf{L}$ . The first rule avoids retaining unnecessarily small elements in the factorization.

### Problems

1. Consider the method of steepest descent and, using (9.3.1) and (9.3.2b), show that

$$E(\mathbf{y}) = \frac{1}{2}(\mathbf{y} - \mathbf{x})^T \mathbf{A}(\mathbf{y} - \mathbf{x}) - \frac{1}{2}\mathbf{x}^T \mathbf{A}\mathbf{x}.$$

2. Show that the SSOR procedure (9.3.25) has the form of (9.3.21) with  $\hat{\mathbf{M}}$  and  $\hat{\mathbf{N}}$  given by (9.3.26).
3. Indicate the algorithm for solving (9.3.27c - 9.3.27e) when  $\mathbf{D}$  and  $\mathbf{L}$  correspond to centered differencing of the two-dimensional Laplacian.

## 9.4 Krylov Subspace Methods

The conjugate gradient method is applicable to symmetric and positive definite linear systems. While many elliptic problems satisfy these conditions, there are important

problems that do not. Convection-diffusion equations, studied in Section 4.4 are a good example. The use of upwind differencing of the convective (first-derivative) terms will produce an unsymmetric algebraic system. Of course, if our goal is to solve (9.1.1), we could modify the problem to

$$\mathbf{A}^T \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{b}. \quad (9.4.1)$$

This system is symmetric and positive definite; hence, we can solve it using the steepest descent or conjugate gradient method. However, the condition number (9.3.4c) of  $\mathbf{A}^T \mathbf{A}$  is the square of that of  $\mathbf{A}$ , and this may lead to ill conditioning. Thus, we can expect slow convergence when applying this procedure.

An alternate possibility is to seek an approximate solution by *projection methods*. With such an approach, our goal is to find approximations

$$\tilde{\mathbf{x}} = \mathbf{x}^{(0)} + \mathbf{v} \quad (9.4.2a)$$

where  $\mathbf{x}^{(0)}$  is an initial guess and  $\mathbf{v}$  is an element of an  $M$ -dimensional ( $M \leq N$ ) subspace  $\mathcal{V}$  of  $\mathfrak{R}^N$ . The subspace  $\mathcal{V}$  is called the *trial*, *candidate*, or *search space*. In order to determine  $\tilde{\mathbf{x}}$  we must specify  $M$  conditions or constraints for  $\mathbf{v}$  to satisfy. Using the *Petrov-Galerkin method*, we introduce a second  $M$ -dimensional subspace  $\mathcal{W}$  of  $\mathfrak{R}^N$ , and determine  $\tilde{\mathbf{x}}$  such that

$$(\tilde{\mathbf{r}}, \mathbf{w}) = (\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}, \mathbf{w}) = 0, \quad \forall \mathbf{w} \in \mathcal{W}. \quad (9.4.2b)$$

Thus, the residual  $\mathbf{r}$  is orthogonal to all ( $M$  linearly independent) vectors  $\mathbf{w} \in \mathcal{W}$ . The subspace  $\mathcal{W}$  is called the *test* or *constraint space*. Orthogonality can involve the usual  $\mathcal{L}^2$  inner product

$$(\mathbf{v}, \mathbf{w}) := \mathbf{w}^T \mathbf{v}; \quad (9.4.2c)$$

however, we'll consider other inner products (*e.g.*, strain energy (9.3.12a)) as well.

Using (9.4.2a) in (9.4.2b)

$$(\mathbf{b} - \mathbf{A}(\mathbf{x}^{(0)} + \mathbf{v}), \mathbf{w}) = (\mathbf{r}^{(0)} - \mathbf{A}\mathbf{v}, \mathbf{w}) = 0, \quad \forall \mathbf{w} \in \mathcal{W}. \quad (9.4.3)$$

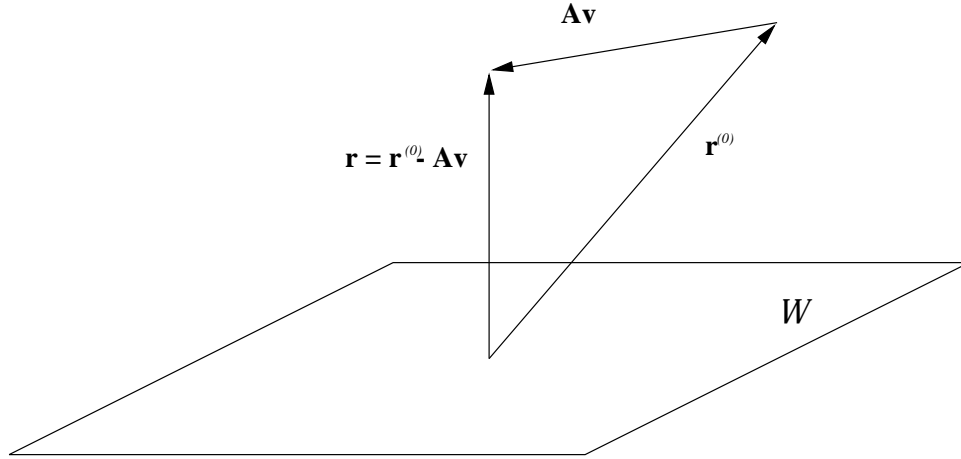


Figure 9.4.1: Orthogonal projection in three dimensions.

The orthogonality condition is now imposed on the “new residual”  $\mathbf{r}^{(0)} - \mathbf{A}\mathbf{v}$  and all vectors  $\mathbf{w} \in \mathcal{W}$ . This is illustrated for two- and three-dimensional spaces in Figure 9.4.1.

Two choices of the test space  $\mathcal{W}$  are important:  $\mathcal{W} = \mathbf{A}\mathcal{V}$  and  $\mathcal{W} = \mathcal{V}$ . In the first case, the solution of (9.4.2b) (or (9.4.3)) is optimal in the sense of minimizing  $\|\tilde{\mathbf{r}}\|_2$ ,  $\forall \mathbf{w} \in \mathbf{A}\mathcal{V}$ . To show this, let  $\tilde{\mathbf{x}}$  satisfy (9.4.2b), add a perturbation  $\delta\mathbf{x}$ , and consider the residual

$$\|\mathbf{r}\|_2^2 = (\mathbf{b} - \mathbf{A}(\tilde{\mathbf{x}} + \delta\mathbf{x}))^T (\mathbf{b} - \mathbf{A}(\tilde{\mathbf{x}} + \delta\mathbf{x})).$$

Expanding

$$\|\mathbf{r}\|_2^2 = \|\tilde{\mathbf{r}}\|_2^2 - 2(\mathbf{A}\delta\mathbf{x})^T (\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}) + \delta\mathbf{x}\mathbf{A}^T \mathbf{A}\delta\mathbf{x}.$$

With  $\delta\mathbf{x} \in \mathcal{V}$  and  $\mathbf{w} = \mathbf{A}\delta\mathbf{x} \in \mathcal{W}$ , we have from (9.4.2b)

$$(\mathbf{A}\delta\mathbf{x})^T (\mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}) = 0. \quad (9.4.4)$$

Thus,

$$\|\mathbf{r}\|_2^2 = \|\tilde{\mathbf{r}}\|_2^2 + \delta\mathbf{x}\mathbf{A}^T \mathbf{A}\delta\mathbf{x}.$$

Since  $\delta\mathbf{x}\mathbf{A}^T \mathbf{A}\delta\mathbf{x}$  is non-negative definite, we establish  $\|\tilde{\mathbf{r}}\|_2^2$  as the minimum. The converse, *i.e.*, that minimizers of  $\|\mathbf{r}\|$  satisfy (9.4.2b), is also true and its proof follows similar arguments.

Using (9.4.2)

$$\tilde{\mathbf{r}} = \mathbf{b} - \mathbf{A}(\mathbf{x}^{(0)} - \mathbf{v}) = \mathbf{r}^{(0)} - \mathbf{A}\mathbf{v}.$$

Condition (9.4.4) enforces orthogonality between  $\tilde{\mathbf{r}}$  and all vectors in  $\mathbf{A}\mathcal{V}$ . Using the above relation, we say that  $\mathbf{A}\mathbf{v}$  is the *orthogonal projection* of  $\mathbf{r}^{(0)}$  onto the subspace  $\mathcal{V}$  (Figure 9.4.1). Methods of this type are called *residual projection methods*.

When  $\mathcal{W} = \mathcal{V}$ , the method (9.4.2b) is called *Galerkin's method*. When  $\mathbf{A}$  is symmetric and positive definite, Galerkin's method is optimal in the sense of minimizing

$$\|\mathbf{x} - \mathbf{y}\|_A^2 = (\mathbf{x} - \mathbf{y})^T \mathbf{A}(\mathbf{x} - \mathbf{y}) \quad (9.4.5a)$$

for all  $\mathbf{y} = \mathbf{x}^{(0)} + \mathbf{v}$ ,  $\mathbf{v} \in \mathcal{V}$ . The appropriate inner product for use with this strain energy norm is

$$(\mathbf{v}, \mathbf{w}) := \mathbf{w}^T \mathbf{A} \mathbf{v}. \quad (9.4.5b)$$

Thus, if  $\tilde{\mathbf{x}}$  satisfies (9.4.2b), we have

$$\mathbf{w}^T \mathbf{A}^T (\mathbf{b} - \mathbf{A} \tilde{\mathbf{x}}) = \mathbf{0}, \quad \forall \mathbf{w} \in \mathcal{V}. \quad (9.4.5c)$$

Proving that  $\tilde{\mathbf{x}}$  also minimizes (9.4.5a), and conversely, follows the lines of the proof for residual projection methods and is left as an exercise (*cf.* Problem 1 at the end of this section).

If our aim is to address nonsymmetric systems, we will primarily be interested in residual projection methods; however, let us proceed in a more general manner for the moment and examine solution schemes for (9.4.2b). Let

$$\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M], \quad \mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \dots, \mathbf{w}_M] \quad (9.4.6a)$$

be  $N \times M$  matrices whose columns form bases for  $\mathcal{V}$  and  $\mathcal{W}$ , respectively. Then, letting

$$\mathbf{v} = \mathbf{V} \mathbf{c}, \quad \mathbf{w} = \mathbf{W} \mathbf{d}, \quad (9.4.7)$$

we write (9.4.3) as

$$(\mathbf{r}^{(0)} - \mathbf{A} \mathbf{V} \mathbf{c}, \mathbf{W} \mathbf{d}) = 0.$$

The requirement that this result hold for all  $\mathbf{w} \in \mathcal{W}$  is equivalent to it holding for all possible choices of  $\mathbf{d}$ . For this to occur, we must have

$$(\mathbf{r}^{(0)} - \mathbf{A} \mathbf{V} \mathbf{c}, \mathbf{W}) = \mathbf{0}.$$

When the inner product corresponds to (9.4.2c), we have

$$\mathbf{W}^T(\mathbf{r}^{(0)} - \mathbf{A}\mathbf{V}\mathbf{c}) = \mathbf{0}.$$

Thus,  $\mathbf{c}$  satisfies

$$\mathbf{W}^T \mathbf{A} \mathbf{V} \mathbf{c} = \mathbf{W}^T \mathbf{r}^{(0)}.$$

Assuming that the  $M \times M$  matrix  $\mathbf{W}^T \mathbf{A} \mathbf{V}$  is invertible, we find

$$\mathbf{c} = (\mathbf{W}^T \mathbf{A} \mathbf{V})^{-1} \mathbf{W}^T \mathbf{r}^{(0)} \quad (9.4.8a)$$

and, using (9.4.2a) and (9.3.5c),

$$\tilde{\mathbf{x}} = \mathbf{x}^{(0)} + \mathbf{V}(\mathbf{W}^T \mathbf{A} \mathbf{V})^{-1} \mathbf{W}^T \mathbf{r}^{(0)}. \quad (9.4.8b)$$

Let's look at two methods where the subspaces  $\mathcal{V}$  and  $\mathcal{W}$  are one dimensional.

*Example 9.4.1.* Let  $\mathcal{V} = \mathcal{W} = \text{span}\{\mathbf{v}_1\}$  where  $\mathbf{v}_1$  is an  $N$ -vector and choose  $\mathbf{v}_1 = \mathbf{r}^{(0)}$ .

Then, using (9.4.8)

$$\mathbf{c} = \frac{(\mathbf{r}^{(0)})^T \mathbf{r}^{(0)}}{(\mathbf{r}^{(0)})^T \mathbf{A} \mathbf{r}^{(0)}}$$

and

$$\tilde{\mathbf{x}} = \mathbf{x}^{(0)} + c\mathbf{r}^{(0)}.$$

We recognize this as being one step of the steepest descent algorithm (Figure 9.3.1).

*Example 9.4.2.* In minimum residual (MR) iteration, we select  $\mathcal{V} = \text{span}\{\mathbf{v}_1\} = \mathbf{r}^{(0)}$  and  $\mathcal{W} = \text{span}\{\mathbf{w}_1\} = \mathbf{A}\mathbf{r}^{(0)}$ . Using (9.4.8)

$$c = \frac{(\mathbf{r}^{(0)})^T \mathbf{A}^T \mathbf{r}^{(0)}}{(\mathbf{r}^{(0)})^T \mathbf{A}^T \mathbf{A} \mathbf{r}^{(0)}}$$

and

$$\tilde{\mathbf{x}} = \mathbf{x}^{(0)} + c\mathbf{r}^{(0)}.$$

The matrix  $\mathbf{A}$  need not be symmetric but only be positive definite for MR to converge ([4], Section 5.3).

Many iterative techniques for symmetric and non-symmetric matrices utilize Krylov subspaces for the trial space  $\mathcal{V}$ . A *Krylov subspace* of dimension  $M$  is

$$\mathcal{K}_M(\mathbf{A}, \mathbf{r}^{(0)}) = \text{span}\{\mathbf{r}^{(0)}, \mathbf{A}\mathbf{r}^{(0)}, \mathbf{A}^2\mathbf{r}^{(0)}, \dots, \mathbf{A}^{M-1}\mathbf{r}^{(0)}\}. \quad (9.4.9)$$

It seems clear that vectors  $\mathbf{v} \in \mathcal{K}_M$  have the form  $\mathbf{v} = p(\mathbf{A})\mathbf{r}^{(0)}$  where  $p(\mathbf{A})$  is a polynomial of degree not exceeding  $M - 1$  in  $\mathbf{A}$ . Methods using Krylov subspaces as trial spaces are called *Krylov subspace methods* or simply *Krylov methods*. Different approaches are distinguished by their choice of the test space  $\mathcal{W}$ , with the two most popular choices being  $\mathcal{K}_M$  and  $\mathbf{A}\mathcal{K}_M$ . The *generalized minimal residual* (GMRES) method, in particular, uses the latter choice. Thus, if we define the Krylov matrix

$$\mathbf{K}_M = [\mathbf{r}^{(0)}, \mathbf{A}\mathbf{r}^{(0)}, \mathbf{A}^2\mathbf{r}^{(0)}, \dots, \mathbf{A}^{M-1}\mathbf{r}^{(0)}],$$

we could select

$$\mathbf{V}_M = \mathbf{K}_M, \quad \mathbf{W}_M = \mathbf{A}\mathbf{K}_M,$$

choose the  $M$ th solution iterate according to (9.4.2a) and (9.3.5c) as

$$\mathbf{x}^{(M)} = \mathbf{x}^{(0)} + \mathbf{V}_M\mathbf{c}^{(M)}, \quad (9.4.10)$$

and determine the  $M$ th-GMRES iterate from (9.4.8a) as

$$\mathbf{c}^{(M)} = (\mathbf{K}_M^T \mathbf{A}^T \mathbf{A} \mathbf{K}_M)^{-1} \mathbf{K}_M^T \mathbf{A}^T \mathbf{r}^{(0)}.$$

We see that:

1. The GMRES procedure is the extension of the MR procedure to higher-dimensional spaces.
2. The dimension of the Krylov space increases by one after each iteration.
3. As may be expected from our earlier work,  $\mathbf{x}^{(M)}$  minimizes

$$\|\mathbf{r}^{(M)}\|_2 = \|\mathbf{b} - \mathbf{A}\mathbf{x}^{(M)}\|_2$$

Unfortunately, the procedure just outlined is unstable (sensitive to round-off error accumulation). It will be necessary to select a more well conditioned basis for  $\mathcal{K}_M$  and we'll do this by constructing a set of mutually orthogonal vectors. This orthogonal projection onto  $\mathcal{K}_M$  is usually done by Arnoldi's method and a pseudocode procedure for its construction appears in Figure 9.4.2. The procedure begins with an initial vector  $\mathbf{v}_1$

which is the initial residual  $\mathbf{r}^{(0)}$  normalized to have a unit Euclidean length. Successive vectors  $\mathbf{v}_j$  are chosen to be orthogonal to the previous ones and to have unit lengths. Indeed, the algorithm of Figure 9.4.2 is just classical Gram-Schmidt orthogonalization modified by the presence of  $\mathbf{A}$  to produce an orthonormal basis for the Krylov subspace

$$\mathcal{K}_M(\mathbf{A}, \mathbf{v}_1) = \text{span}\{\mathbf{v}_1, \mathbf{A}\mathbf{v}_1, \mathbf{A}^2\mathbf{v}_1, \dots, \mathbf{A}^{M-1}\mathbf{v}_1\}. \quad (9.4.11)$$

The algorithm will halt prematurely if any unnormalized vector  $\mathbf{w}_j$  has a zero Euclidean length. The Boolean variable *quit* is set to **true** should this happen. Proving these assertions may be done with an induction argument ([4], Section 6.3), but we'll proceed less formally. At the first ( $j = 1$ ) step of the algorithm, we obtain

```

procedure arnoldi
  quit = ( $\|\mathbf{r}^{(0)}\|_2 = 0$ )
  if not quit then
     $\mathbf{v}_1 = \mathbf{r}^{(0)} / \|\mathbf{r}^{(0)}\|_2$ 
  end if
   $j = 1$ 
  while ( $j \leq M$ ) and (not quit) do
     $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$ 
    for  $i = 1$  to  $j$  do
       $h_{ij} = \mathbf{v}_i^T \mathbf{A}\mathbf{v}_j$ 
       $\mathbf{w}_j = \mathbf{w}_j - h_{ij}\mathbf{v}_i$ 
    end for
     $h_{j+1,j} = \|\mathbf{w}_j\|_2$ 
    quit = ( $h_{j+1,j} = 0$ )
    if not quit then
       $\mathbf{v}_{j+1} = \mathbf{w}_j / h_{j+1,j}$ 
    end if
     $j = j + 1$ 
  end while

```

Figure 9.4.2: Arnoldi Gram-Schmidt orthogonal basis construction for  $\mathcal{K}_M$ .

$$h_{21}\mathbf{v}_2 = \mathbf{w}_1 = \mathbf{A}\mathbf{v}_1 - h_{11}\mathbf{v}_1.$$

Taking an inner product with  $\mathbf{v}_1$

$$h_{21}\mathbf{v}_1^T \mathbf{v}_2 = \mathbf{v}_1^T \mathbf{A}\mathbf{v}_1 - h_{11}\mathbf{v}_1^T \mathbf{v}_1.$$

However,  $\mathbf{v}_1^T \mathbf{v}_1 = 1$  and  $h_{11} = \mathbf{v}_1^T \mathbf{A} \mathbf{v}_1$ , so  $\mathbf{v}_1$  is orthogonal to  $\mathbf{v}_2$ . It is also clear that  $\mathbf{v}_2$  is a linear combination of  $\mathbf{v}_1$  and  $\mathbf{A} \mathbf{v}_1$  and, hence, an element of  $\mathcal{K}_2(\mathbf{A}, \mathbf{v}_1)$ . The extension of the induction argument to higher values of  $j$  is similar.

From the **for** loop within Arnoldi's algorithm, we may infer

$$h_{j+1,j} \mathbf{v}_{j+1} = \mathbf{A} \mathbf{v}_j - \sum_{i=1}^j h_{ij} \mathbf{v}_i, \quad j = 1, 2, \dots, M, \quad (9.4.12a)$$

or

$$\mathbf{A} \mathbf{v}_j = \sum_{i=1}^{j+1} h_{ij} \mathbf{v}_i, \quad j = 1, 2, \dots, M. \quad (9.4.12b)$$

Let  $\mathbf{V}_M$  be the  $N \times M$  matrix whose columns are  $\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_M$  and let  $\bar{\mathbf{H}}_M$  be the  $(M+1) \times M$  Hessenberg matrix

$$\bar{\mathbf{H}}_M = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{1M} \\ h_{21} & h_{22} & \cdots & h_{2M} \\ & h_{32} & \cdots & h_{3M} \\ & & \ddots & \vdots \\ & & & h_{M+1,M} \end{bmatrix}. \quad (9.4.13)$$

Then (9.4.12) can be written in matrix form as

$$\mathbf{A} \mathbf{V}_M = \mathbf{V}_{M+1} \bar{\mathbf{H}}_M. \quad (9.4.14a)$$

Additionally, let  $\mathbf{H}_M$  be the matrix obtained by deleting the last row of  $\bar{\mathbf{H}}_M$ . Then, either by using (9.4.12a) or (9.4.14a), we have

$$\mathbf{A} \mathbf{V}_M = \mathbf{V}_M \mathbf{H}_M + \mathbf{w}_M \mathbf{e}_M^T \quad (9.4.14b)$$

where, from Arnoldi's algorithm,  $\mathbf{w}_M = h_{M+1,M} \mathbf{v}_{M+1}$  and  $\mathbf{e}_M$  is the  $M$ th column of the identity matrix. Finally, multiplying (9.4.14b) by  $\mathbf{V}_M^T$  and using the orthogonality of its columns, we obtain

$$\mathbf{V}_M^T \mathbf{A} \mathbf{V}_M = \mathbf{H}_M. \quad (9.4.14c)$$

The Arnoldi algorithm as stated in Figure 9.4.2 is subject to round-off error accumulation. The modified Gram-Schmidt procedure, as illustrated in Figure 9.4.3, is much

more stable and less sensitive to round-off error difficulties. Mathematically, the steps in the two Arnoldi procedures are identical; however, the modified Gram-Schmidt version avoids cancellations of nearly equal quantities. In cases of severe round-off error accumulation, Householder transformations can be used to construct the orthogonal basis. We will not do this here, but interested readers may consult Saad [4], Section 6.3.

```

procedure marnoldi
  quit = ( $\|\mathbf{r}^{(0)}\|_2 = 0$ )
  if not quit then
     $\mathbf{v}_1 = \mathbf{r}^{(0)} / \|\mathbf{r}^{(0)}\|_2$ 
  end if
   $j = 1$ 
  while ( $j \leq M$ ) and (not quit) do
     $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$ 
    for  $i = 1$  to  $j$  do
       $h_{ij} = \mathbf{v}_i^T \mathbf{w}_j$ 
       $\mathbf{w}_j = \mathbf{w}_j - h_{ij}\mathbf{v}_i$ 
    end for
     $h_{j+1,j} = \|\mathbf{w}_j\|_2$ 
    quit = ( $h_{j+1,j} = 0$ )
    if not quit then
       $\mathbf{v}_{j+1} = \mathbf{w}_j / h_{j+1,j}$ 
    end if
     $j = j + 1$ 
  end while

```

Figure 9.4.3: Arnoldi modified Gram-Schmidt orthogonal basis construction for  $\mathcal{K}_M$ .

We are now in a position to improve the stability of the GMRES procedure. Again, choose  $\mathbf{x}^{(M)}$  according to (9.4.10) and calculate the residual

$$\mathbf{b} - \mathbf{A}\mathbf{x}^{(M)} = \mathbf{b} - \mathbf{A}(\mathbf{x}^{(0)} - \mathbf{V}_M\mathbf{c}^{(M)}) = \mathbf{r}^{(0)} - \mathbf{A}\mathbf{V}_M\mathbf{c}^{(M)}.$$

Using (9.4.14)

$$\mathbf{b} - \mathbf{A}\mathbf{x}^{(M)} = \beta\mathbf{v}_1 - \mathbf{V}_{M+1}\bar{\mathbf{H}}_M\mathbf{c}^{(M)} = \mathbf{V}_{M+1}(\beta\mathbf{e}_1 - \bar{\mathbf{H}}_M\mathbf{c}^{(M)}) \quad (9.4.15a)$$

where  $\mathbf{e}_1$  is the first column of the identity matrix and (Figure 9.4.3)

$$\beta = \|\mathbf{r}^{(0)}\|_2. \quad (9.4.15b)$$

Since the GMRES approximation minimizes  $\|\mathbf{b} - \mathbf{A}\mathbf{x}^{(M)}\|_2$ , our task is to find  $\mathbf{c}^{(M)}$  as the minimizer of  $\|\mathbf{V}_{M+1}(\beta\mathbf{e}_1 - \bar{\mathbf{H}}_M\mathbf{c}^{(M)})\|_2$ . Actually, since  $\mathbf{V}_M$  is orthogonal, it suffices to minimize  $\|\beta\mathbf{e}_1 - \bar{\mathbf{H}}_M\mathbf{c}^{(M)}\|_2$ . The approximate solution is then given by (9.4.10).

The minimizer  $\mathbf{c}^{(M)}$  is computationally inexpensive. It requires the solution of an  $(M + 1) \times M$  least-squares problem where  $M$  is typically small relative to  $N$ . An algorithm for the GMRES procedure appears in Figure 9.4.4. The procedure is basically the Arnoldi-modified Gram-Schmidt algorithm with a least-squares procedure. Some additional comments follow.

**procedure** GMRES

$\bar{\mathbf{H}}_M = \mathbf{0}$

$\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\mathbf{x}^{(0)}$

$\beta = \|\mathbf{r}^{(0)}\|_2$

*quit* = ( $\beta = 0$ )

**if not** *quit* **then**

$\mathbf{v}_1 = \mathbf{r}^{(0)}/\beta$

**end if**

$j = 1$

**while** ( $j \leq M$ ) **and** (**not** *quit*) **do**

$\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$

**for**  $i = 1$  **to**  $j$  **do**

$h_{ij} = \mathbf{v}_i^T \mathbf{w}_j$

$\mathbf{w}_j = \mathbf{w}_j - h_{ij}\mathbf{v}_i$

**end for**

$h_{j+1,j} = \|\mathbf{w}_j\|_2$

*quit* = ( $h_{j+1,j} = 0$ )

**if not** *quit* **then**

$\mathbf{v}_{j+1} = \mathbf{w}_j/h_{j+1,j}$

**end if**

$j = j + 1$

**end while**

$m = j - 1$

Determine  $\mathbf{c}^{(M)}$  as the minimizer of  $\|\beta\mathbf{e}_1 - \bar{\mathbf{H}}_M\mathbf{c}^{(M)}\|_2$

$\mathbf{x}^{(M)} = \mathbf{x}^{(0)} + \mathbf{V}_M\mathbf{c}^{(M)}$

Figure 9.4.4: Generalized minimum residual algorithm.

1. As stated, the GMRES procedure does not calculate the solution at each step. Thus, it is difficult to know when to stop. It would be better to calculate  $\mathbf{x}^{(j)}$



We multiply  $\beta \mathbf{e}_1 - \bar{\mathbf{H}}_M \mathbf{c}$  on the left by a sequence of rotations  $\mathbf{\Omega}_i$  with the coefficients  $c_i$  and  $s_i$  designed to eliminate  $h_{i+1,i}$ ,  $i = 1, 2, \dots, M$ . If  $M$  were five, we would have

$$\bar{\mathbf{H}}_M = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{14} & h_{15} \\ h_{21} & h_{22} & h_{23} & h_{24} & h_{25} \\ & h_{32} & h_{33} & h_{34} & h_{35} \\ & & h_{43} & h_{44} & h_{45} \\ & & & h_{54} & h_{55} \\ & & & & h_{65} \end{bmatrix}, \quad \bar{\mathbf{g}} = \begin{bmatrix} \beta \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

We multiply  $\bar{\mathbf{H}}_M$  and  $\bar{\mathbf{g}}$  by

$$\mathbf{\Omega}_1 = \begin{bmatrix} c_1 & s_1 & & & \\ -s_1 & c_1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & & 1 \end{bmatrix}$$

with

$$c_1 = \frac{h_{11}}{\sqrt{h_{11}^2 + h_{21}^2}} \quad s_1 = \frac{h_{21}}{\sqrt{h_{11}^2 + h_{21}^2}}$$

to obtain

$$\bar{\mathbf{H}}_M^{(1)} = \begin{bmatrix} h_{11}^{(1)} & h_{12}^{(1)} & h_{13}^{(1)} & h_{14}^{(1)} & h_{15}^{(1)} \\ & h_{22}^{(1)} & h_{23}^{(1)} & h_{24}^{(1)} & h_{25}^{(1)} \\ & & h_{33} & h_{34} & h_{35} \\ & & & h_{43} & h_{44} & h_{45} \\ & & & & h_{54} & h_{55} \\ & & & & & h_{65} \end{bmatrix}, \quad \bar{\mathbf{g}}^{(1)} = \begin{bmatrix} c_1 \beta \\ -s_1 \beta \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

The process continues with the parameters for rotation  $i$  satisfying

$$c_i = \frac{h_{ii}^{(i-1)}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}} \quad s_i = \frac{h_{i+1,i}}{\sqrt{(h_{ii}^{(i-1)})^2 + h_{i+1,i}^2}}. \quad (9.4.16c)$$

At the end of  $M$  ( $= 5$ ) rotations, we have

$$\bar{\mathbf{H}}_M^{(5)} = \begin{bmatrix} h_{11}^{(5)} & h_{12}^{(5)} & h_{13}^{(5)} & h_{14}^{(5)} & h_{15}^{(5)} \\ & h_{22}^{(5)} & h_{23}^{(5)} & h_{24}^{(5)} & h_{25}^{(5)} \\ & & h_{33}^{(5)} & h_{34}^{(5)} & h_{35}^{(5)} \\ & & & h_{44}^{(5)} & h_{45}^{(5)} \\ & & & & h_{55}^{(5)} \\ & & & & & 0 \end{bmatrix}, \quad \bar{\mathbf{g}}^{(5)} = \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_6 \end{bmatrix}.$$

In the general case, let

$$\mathbf{Q}_M = \mathbf{\Omega}_M \mathbf{\Omega}_{M-1} \dots \mathbf{\Omega}_1 \quad (9.4.17a)$$

and

$$\bar{\mathbf{R}}_M = \bar{\mathbf{H}}_M^{(M)} = \mathbf{Q}_M \bar{\mathbf{H}}_M, \quad \bar{\mathbf{g}}_M = \bar{\mathbf{g}}_M^{(M)} = \mathbf{Q}_M \beta \mathbf{e}_1. \quad (9.4.17b)$$

Since the matrices  $\mathbf{\Omega}_i$ ,  $i = 1, 2, \dots, M$ , and  $\mathbf{Q}_M$  are orthogonal,

$$\min_{\mathbf{c}} \|\beta \mathbf{e}_1 - \bar{\mathbf{H}}_M \mathbf{c}\|_2 = \min_{\mathbf{c}} \|\bar{\mathbf{g}}_M - \bar{\mathbf{R}}_M \mathbf{c}\|_2.$$

The solution of this problem is obtained by deleting the last row of  $\bar{\mathbf{R}}_M$  and  $\bar{\mathbf{g}}_M$  and solving the resulting upper triangular linear system for  $\mathbf{c}^{(M)}$ . Thus, let  $\mathbf{R}_M$  be the  $M \times M$  matrix obtained by deleting the last row of  $\bar{\mathbf{R}}_M$  and  $\mathbf{g}_M$  be the  $M$ -vector obtained by deleting the last element of  $\bar{\mathbf{g}}_M$ , then

$$\mathbf{c}^{(M)} = \mathbf{R}_M^{-1} \mathbf{g}_M. \quad (9.4.18)$$

*Example 9.4.3.* Saad [4] solves the partial differential equation

$$-u_{xx} - u_{yy} - u_{zz} + 10(e^{xy}u)_x + 10(e^{-xy}u)_y = f(x, y, z)$$

on a unit square with trivial Dirichlet boundary conditions. He chooses a  $17 \times 17 \times 17$  mesh and uses centered differencing. The function  $f(x, y, z)$  is chosen so that the right side  $\mathbf{b}$  of the discrete problem (9.1.1) satisfies

$$\mathbf{b} = \mathbf{A}[1, 1, \dots, 1]^T.$$

With Dirichlet boundary conditions, the dimension of the discrete system is  $(16)^3 = 4096$ . Saad [4] solves this system by GMRES procedures with and without preconditioning. The results of some of his calculations are reported in Table 9.4. The column labeled *Iter* reports the number of matrix-by-vector multiplications to reduce the initial residual by a factor of  $10^7$  in the  $\mathcal{L}^2$  norm. The parameter *Kflops* reports the number of floating point operations performed divided by 1000. The parameters *Residual* and *Error* record the residual and error in the  $\mathcal{L}^2$  norm. The first row of Table 9.4 reports data for restarted GMRES with a Krylov space of dimension  $M = 10$ . The second row displays data for GMRES calculations performed with a truncated orthogonalization using  $k = 10$  columns. The third and fourth rows contain results of preconditioned GMRES calculations using, respectively, SSOR and ILU(0) preconditioners. The acceleration parameter

$\omega$  was set to unity for the SSOR technique. The preconditioned versions increased the dimension of the Krylov space until convergence was attained. All procedures were initiated with a random initial guess.

Method	Iter	Kflops	Residual	Error
Restarted GMRES	67	11862	0.37(-3)	0.28(-3)
Truncated GMRES	75	22798	0.64(-3)	0.32(-3)
SSOR-GMRES	20	4870	0.14(-2)	0.30(-3)
ILU(0)-GMRES	17	4004	0.52(-3)	0.30(-3)

Table 9.4.1: Number of iterations (Iter) to convergence, number of floating point operations (Kflops), and the residual and error in  $\mathcal{L}^2$  for GMRES procedures with and without preconditioners (Example 9.4.3).

The results with preconditioning are much better than those without. The restarted and incomplete orthogonalization versions of GMRES have comparable performance. Likewise, the SSOR and ILU(0) preconditionings are comparable.

### Problems

1. With  $\mathbf{A}$  being positive definite and symmetric, show that solutions  $\tilde{\mathbf{x}}$  of (9.4.5c) minimizes (9.4.5a), and conversely.



# Bibliography

- [1] J.A. George. Nested dissection of a regular finite element mesh. *SIAM Journal on Numerical Analysis*, 10:345–363, 1973.
- [2] J.A. George and J.W. Liu. *Computational Solution of Large Sparse Positive Definite Systems*. Prentice-Hall Series in Automatic Computation. Prentice-Hall, Englewood Cliffs, 1981.
- [3] E. Isaacson and H.B. Keller. *Analysis of Numerical Methods*. John Wiley and Sons, New York, 1966.
- [4] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS, Boston, 1996.
- [5] J.C. Strikwerda. *Finite Difference Schemes and Partial Differential Equations*. Chapman and Hall, Pacific Grove, 1989.
- [6] R.S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, 1963.
- [7] E. L. Wachpress. *Iterative Solution of Elliptic Systems*. Prentice-Hall, Englewood Cliffs, 1966.