Chapter 20

Basic Iterative Methods

You should be familiar with

- Eigenvalues and the spectral radius
- Matrix norms
- Finite difference methods for approximating derivatives
- Block matrices

When using finite difference methods for the solution of partial differential equations, the matrix may be extremely large, possibly 1000×1000 or larger. The main source of large matrix problems is the discretization of partial differential equations, but large linear systems also arise in other applications such as the design and computer analysis of circuits, and chemical engineering processes. Often these matrices are *sparse*, which means that most of the matrix entries are 0. Standard Gaussian elimination turns zeros into nonzeros, reducing the sparsity. For large, sparse matrices, the preferred method of solution is to use an *iterative method*. An iterative method generates a sequence that converges to the solution, and the iteration is continued until a desired error tolerance is satisfied. Unlike Gaussian elimination, iterative methods do not alter the matrix, but use only a small set of vectors obtained from the matrix, so they use far less storage than working directly with the matrix. This chapter presents three classical iterative methods, the Jacobi, Gauss-Seidel, and the successive overrelaxation (SOR) iterations. These methods are relatively slow, but they provide an introduction to the idea of using iteration to solve systems and form a basis for more sophisticated methods such as multigrid. The main idea of multigrid is to accelerate the convergence of a basic iterative method by solving a coarse problem. It takes an $n \times n$ grid and uses every other point on an $n/2 \times n/2$ grid to estimate values on the larger grid using interpolation. Points on an $n/4 \times n/4$ grid are used to approximate values on the $n/2$ grid, and so forth, forming a recursive algorithm. These methods are beyond the scope of this book. The interested reader can consult Refs. [1, 63] for detailed information about these iterative methods.

20.1 JACOBI METHOD

For the Jacobi method, write each unknown in terms of the other unknowns. We illustrate this process using a 4×4 system.

$$
\begin{aligned}
a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + a_{14}x_4 = b_1 &\Longrightarrow x_1 = [b_1 - (a_{12}x_2 + a_{13}x_3 + a_{14}x_4)]/a_{11} \\
a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + a_{24}x_4 = b_2 &\Longrightarrow x_2 = [b_2 - (a_{21}x_1 + a_{23}x_3 + a_{24}x_4)]/a_{22} \\
a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + a_{34}x_4 = b_3 &\Longrightarrow x_3 = [b_3 - (a_{31}x_1 + a_{32}x_2 + a_{34}x_4)]/a_{33} \\
a_{41}x_1 + a_{42}x_2 + a_{43}x_3 + a_{44}x_4 = b_4 &\Longrightarrow x_4 = [b_4 - (a_{41}x_1 + a_{42}x_2 + a_{43}x_3)]/a_{44}
\end{aligned}
$$

Assume initial values for x_1, x_2, x_3, and x_4, insert them in the right-hand side of the equations and compute a second set of approximate values for the unknowns. These new values are then substituted into the right-hand side of the equations to obtain a third set of approximate solutions, and so forth, until the iterations produce a relative error estimate that is small enough. For the general case of n unknowns, the iteration is defined by

$$
x_i = \frac{1}{a_{ii}} \left[b_i - \left(\sum_{j=1, j \neq i}^{n} a_{ij} x_j \right) \right], \quad 1 \leq i \leq n \tag{20.1}
$$

Definition 20.1. Let x be a vector obtained during an iteration. In order to clearly distinguish the number of the iteration and a component of the vector, we use the notation $x_i^{(k)}$, where k refers to the iteration number and i refers to the ith component of $x^{(k)}$.

Numerical Linear Algebra with Applications. http://dx.doi.org/10.1016/B978-0-12-394435-1.00020-X

If a good initial approximation for the solution $x^{(0)}$ is known, use $x_1^{(0)}, x_2^{(0)}, \ldots, x_n^{(0)}$ to begin the iteration. If there is no estimate for the initial value $x^{(0)}$, use $x^{(0)} = 0$. The second approximation for the solution, $x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}$ is calculated by substituting the first estimate into the right-hand side of Equation 20.1:

$$x_i^{(1)} = \frac{1}{a_{ii}}\left[b_i - \left(\sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(0)}\right)\right], \quad 1 \leq i \leq n$$

In general, the estimate $x^{(k)}$ for the solution is calculated from the estimate x^{k-1} by

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[b_i - \left(\sum_{j=1, j\neq i}^{n} a_{ij}x_j^{(k-1)}\right)\right], \quad 1 \leq i \leq n \tag{20.2}$$

The iterations continue until the error tolerance tol is satisfied. What criterion do we use to determine that the algorithm has met the required tolerance? An obvious choice is $\|Ax_i - b\|_2 < \text{tol}$. However, note that

$$A(x - x_i) = Ax - Ax_i = b - Ax_i,$$

so

$$\|x - x_i\|_2 = \left\|A^{-1}(b - Ax_i)\right\|_2 \leq \left\|A^{-1}\right\|_2 \|b - Ax_i\|_2 \leq \left\|A^{-1}\right\|_2 \text{tol}.$$

If $\left\|A^{-1}\right\|_2$ is large, $\|x - x_i\|$ can be large. The best alternative is to apply a relative error estimate and terminate when

$$\frac{\|b - Ax_i\|_2}{\|b\|} \leq \text{tol}.$$

Thus, terminate the iteration when the current residual relative to b is sufficiently small [16, pp. 335-336]. We will call $\frac{\|b-Ax_i\|_2}{\|b\|}$, the *relative residual*.

Example 20.1. Consider the system $Ax = b$, where $A = \begin{bmatrix} 5 & -1 & 2 \\ -1 & 4 & 1 \\ 1 & 6 & -7 \end{bmatrix}$ and $b = \begin{bmatrix} 1 \\ -2 \\ 5 \end{bmatrix}$. We start with $x^{(0)} = 0$, execute the first two iterations in detail, continue for a total of 12 iterations, and compute the relative residual.

$$x_1^{(1)} = \frac{1}{5}(1) = 0.2000, \quad x_2^{(1)} = \frac{1}{4}(-2) = -0.5000, \quad x_3^{(1)} = -0.7143$$

$$x_1^{(2)} = \frac{1}{5}\left(1 - \frac{1}{2} + \frac{10}{7}\right) = 0.3857, \quad x_2^{(2)} = \frac{1}{4}\left(-2 + \frac{1}{5} + \frac{5}{7}\right) = -0.2714, x_3^{(2)} = -\frac{1}{7}\left(5 - \frac{1}{5} - 6\left(-\frac{1}{2}\right)\right) = -1.1143$$

$$\vdots$$

$$x_1^{(12)} = 0.4838, \quad x_2^{(12)} = -0.1795, \quad x_3^{(12)} = -0.7998$$

$$\frac{\left\|b - Ax^{(12)}\right\|_2}{\|b\|_2} = 1.1116 \times 10^{-3}$$

■

20.2 THE GAUSS-SEIDEL ITERATIVE METHOD

In the *Gauss-Seidel method*, start with approximate values $x_2^{(0)}, \ldots, x_n^{(0)}$ if known; otherwise choose $x^{(0)} = 0$. Use these values to calculate $x_1^{(1)}$. Use $x_1^{(1)}$ and $x_3^{(0)}, \ldots, x_n^{(0)}$ to calculate $x_2^{(1)}$, and so forth. At each step, we are applying new vector component values as soon as we compute them. The hope is that this strategy will improve the convergence rate. Applying this method with Equation 20.1, we have the iteration formula:

$$x_1^{(k)} = \frac{1}{a_{11}}\left[b_1 - \left(\sum_{j=2}^{n} a_{1j}x_j^{(k-1)}\right)\right] \tag{20.3}$$

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[b_i - \left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}\right)\right], \quad i = 2, 3, \ldots, n-1 \tag{20.4}$$

$$x_n^{(k)} = \frac{1}{a_{nn}}\left[b_n - \sum_{j=1}^{n-1} a_{nj}x_j^{(k)}\right] \tag{20.5}$$

Example 20.2. Use the matrix of Example 20.1 and apply the Gauss-Seidel method, with the iteration defined by Equations 20.3–20.5. Begin with $x^{(0)} = 0$, execute the first two iterations in detail, continue for a total of 12 iterations, and compute the relative residual.

$$\begin{aligned}
x_1^{(1)} &= \frac{1}{5}(1) = 0.2000, \quad x_2^{(1)} = \frac{1}{4}\left[-2 - \left(-\frac{1}{5} + (1)\,0\right)\right] = -0.4500, x_3^{(1)} = -\frac{1}{7}\left[5 - \left((1)\,\frac{1}{5} + 6\left(-\frac{9}{20}\right)\right)\right] = -1.0714 \\
x_1^{(2)} &= \frac{1}{5}\left[1 - \left((-1)\left(-\frac{9}{20}\right) + 2\left(-\frac{15}{14}\right)\right)\right] = 0.5386, x_2^{(2)} = \frac{1}{4}\left[-2 - \left((-1)\,\frac{377}{700} + (1)\left(-\frac{15}{14}\right)\right)\right] = -0.0975, \\
& \quad x_3^{(3)} = -\frac{1}{7}\left[5 - \left((1)\left(\frac{377}{700}\right) + 6\left(-\frac{39}{400}\right)\right)\right] = -0.7209 \\
&\vdots \\
x_1^{(12)} &= 0.4837, \quad x_2^{(12)} = -0.1794, \quad x_3^{(12)} = -0.7989 \\
\frac{\left\|b - Ax^{(12)}\right\|_2}{\|b\|_2} &= 2.8183 \times 10^{-7}
\end{aligned}$$

If you compare this result with that of Example 20.1, it is clear that the Gauss-Seidel iteration obtained higher accuracy in the same number of iterations. ■

20.3 THE SOR ITERATION

The SOR method was developed to accelerate the convergence of the Gauss-Seidel method. The idea is to successively form a weighted average between the previously computed value $x_i^{(k-1)}$ and the new value

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[b_i - \left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}\right)\right], \quad i = 1, 2, \ldots, n.$$

Weight the newly computed value by ω and the previous value by $(1-\omega)$. By assuming that $\sum_{j=1}^{i-1} a_{ij}x_j^{(k)}$ and $\sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}$ are ignored when $i = 1$ and n, respectively, we have

$$x_i^{(k)} = \frac{\omega}{a_{ii}}\left[b_i - \left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}\right)\right] + (1-\omega)\,x_i^{(k-1)}, \quad i = 1, 2, \ldots, n \tag{20.6}$$

For this method to provide an improvement over the Gauss-Seidel method, ω must be chosen carefully. It is called the *relaxation parameter*. If $\omega = 1$, the SOR method and the Gauss-Seidel method are identical. If $\omega > 1$, we are said to be using *overrelaxation* and if $\omega < 1$ we are using *underrelaxation*.

Example 20.3. Consider the system with $A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{bmatrix}$ and $b = \begin{bmatrix} -1 & 5 & 7 \end{bmatrix}^{\mathrm{T}}$, whose exact solution is $\begin{bmatrix} -11 & 6 & 4 \end{bmatrix}^{\mathrm{T}}$. Apply SOR with $\omega = 1.1$ in detail for two iterations, continue for a total of 15 iterations, and compute the relative residual. Also show the result of applying the Jacobi and Gauss-Seidel methods for 15 iterations.

$$
\begin{aligned}
x_1^{(1)} &= 1.1\,(-1) = -1.1, \quad x_2^{(1)} = \frac{1.1}{2}\,[5 + 1.1] = 3.3550, \quad x_3^{(1)} = \frac{1.1}{3}\,[7 + 1.1 - 3.3550] = 1.7398 \\
x_1^{(2)} &= 1.1\,[-1 - 3.3550 - 1.7398] + (-0.1)\,(-1.1) = -6.5943 \\
x_2^{(2)} &= \frac{1.1}{2}\,[5 - (-6.5943) - 1.7398] + (-0.1)\,(3.3550) = 5.0845 \\
x_3^{(2)} &= \frac{1.1}{3}\,[7 - (-6.5943) - 5.0845] + (-0.10)\,(1.7398) = 2.9463 \\
&\vdots \\
x_1^{(15)} &= -11.0000, x_2^{(15)} = 6.0000, x_3^{(15)} = 4.0000 \\
\frac{\left\|b - Ax^{(15)}\right\|_2}{\|b\|_2} &= 8.18045 \times 10^{-7}
\end{aligned}
$$

After 15 iterations of Gauss-Seidel, the relative residual is 4.72×10^{-5}. It is interesting to note that the Jacobi iteration yields a relative residual of 3.8521; in fact, it diverges. In Section 20.4, we will see why. ■

The choice of $\omega = 1.1$ in Example 20.3 gives a relative residual of 8.18045×10^{-7}, but with $\omega = 1.2$ the residual is 1.4144×10^{-6}. Clearly the accuracy of SOR depends on ω. Later on in this chapter, we will discuss when it is possible to make an optimal choice for ω.

Algorithm 20.1 presents the SOR iteration algorithm. Implementation of the Jacobi and Gauss-Seidel iterations are included in the book software in the functions `jacobi` and `gausseidel`.

Algorithm 20.1 SOR Iteration

```
function SOR(A,b,x0,ω,tol,maxiter)
  % [ x, iter, relresid ] = sor(A,b,x0,omega,tol,maxiter) computes
  % the solution of Ax=b using the SOR iteration.
  % x0 is the initial approximation, ω is the relaxation parameter,
  % tol is the error tolerance, and maxiter is the maximum number of iterations.
  % x is the approximate solution, and iter is the number of iterations required.
  % iter=-1 if the tolerance was not achieved.
  % relresid is the relative residual obtained by the iteration.

  k=1
  x = x0
  while k ≤ maxiter do
    x1 = (ω/a11) (b1 − A(1, 2 : n)) + (1 − ω) x1
    for i=2:n-1 do
      xi = (ω/aii) (bi − A(i, 1 : i − 1) x(1 : i − 1) − ···
      −A(i, i + 1 : n) x(i + 1 : n) )+ (1 − ω) xi
    end for
    xn = (ω/ann) (bn − A(n, 1 : n − 1) x(1 : n − 1)) + (1 − ω) xn
    relesid = ‖b − Ax‖2 / ‖b‖2
    if relresid <tol then
      iter=k
      return[x, iter, relresid]
    end if
    k=k+1
  end while
  iter=-1
  return[x, iter, relresid]
end function
```

NLALIB: The function `sor` implements Algorithm 20.1.

20.4 CONVERGENCE OF THE BASIC ITERATIVE METHODS

The examples may provide the impression that the Jacobi, Gauss-Seidel, and SOR iterations always converge and that Gauss-Seidel always converges faster than Jacobi. Unfortunately, general statements like this are not true, and we must investigate conditions that will guarantee convergence and enable us to compare convergence rates. For this purpose, we express the iterations in the matrix form

$$x^{(k)} = Bx^{(k-1)} + c,$$

where B is called the *iteration matrix*. In this way, we will have the tools of matrix theory available to us.

20.4.1 Matrix Form of the Jacobi Iteration

In the case of the Jacobi iteration, write the coefficient matrix A as a sum of three matrices, a diagonal matrix, and an upper and lower triangular matrix

$$D = \begin{bmatrix} a_{11} & & 0 \\ & a_{22} & \\ & & \ddots \\ 0 & & a_{nn} \end{bmatrix}, \quad L = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ a_{21} & 0 & & \\ \vdots & & \ddots & \\ a_{n1} & \cdots & a_{n,n-1} & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \ldots & a_{2n} \\ \vdots & & \ddots & \\ 0 & 0 & \cdots & 0 \end{bmatrix}, \tag{20.7}$$

$A = D + L + U$ allows us to write the problem $Ax = b$ in the form

$$Dx + Lx + Ux = b.$$

If we assume that A has no zero entries on its diagonal, then

$$D^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & & 0 \\ & \frac{1}{a_{22}} & \\ & & \ddots \\ 0 & & \frac{1}{a_{nn}} \end{bmatrix}$$

and

$$x = D^{-1}b - D^{-1}(L + U)x.$$

The Jacobi iterations 20.2 can be written in matrix form as

$$x^{(k)} = B_J x^{(k-1)} + c_J,$$

where

$$B_J = -D^{-1}(L + U), \quad c_J = D^{-1}b \tag{20.8}$$

20.4.2 Matrix Form of the Gauss-Seidel Iteration

Formulating the Gauss-Seidel iteration as a matrix problem is more challenging. By assuming that $\sum_{j=1}^{i-1} a_{ij}x_j^{(k)}$ and $\sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}$ are ignored when $i = 1$ and n, respectively, we can write the Gauss-Seidel iteration as

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[b_i - \left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}\right)\right], \quad i = 1, 2, 3, \ldots, n.$$

Rearrange the equation by multiplying both sides by a_{ii} and placing all the (k) terms are on the left-hand side to obtain

$$a_{ii}x_i^{(k)} + \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} = b_i - \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}, \quad i = 1, 2, \ldots, n. \tag{20.9}$$

The left-hand side of Equation 20.9 can be expressed by the matrix $Dx^{(k)} + Lx^{(k)}$, and the right-hand side by $b - Ux^{(k-1)}$ using the matrix definitions in Equation 20.7. We now have the matrix equation

$$(L + D)\, x^{(k)} = -Ux^{(k-1)} + b. \tag{20.10}$$

If we solve the left-hand side of Equation 20.10 for x, we obtain $x^{(k)} = -(L + D)^{-1} Ux^{(k-1)} + (L + D)^{-1} b$, giving us the matrix form

$$x^{(k)} = B_{\text{GS}} x^{(k-1)} + c_{\text{GS}}$$

where

$$B_{\text{GS}} = -(L + D)^{-1} U, \quad c_{\text{GS}} = (L + D)^{-1} b \tag{20.11}$$

20.4.3 Matrix Form for SOR

The SOR iteration 20.6 is

$$x_i^{(k)} = \frac{\omega}{a_{ii}} \left[b_i - \left(\sum_{j=1}^{i-1} a_{ij} x_j^{(k)} + \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} \right) \right] + (1 - \omega)\, x_i^{(k-1)}, \quad i = 1, 2, 3, \ldots, n. \tag{20.12}$$

Rearrange Equation 20.12 by multiplying both sides by a_{ii} and placing all the k terms on the left-hand side to obtain

$$a_{ii} x_i^{(k)} + \omega \sum_{j=1}^{i-1} a_{ij} x_j^{(k)} = \omega b_i - \omega \sum_{j=i+1}^{n} a_{ij} x_j^{(k-1)} + (1 - \omega)\, a_{ii} x_i^{(k-1)}, \quad i = 1, 2, \ldots, n. \tag{20.13}$$

Using the matrix definitions in Equation 20.7, we can now express Equation 20.13 in the matrix form

$$(D + \omega L)\, x^{(k)} = ((1 - \omega)\, D - \omega U)\, x^{(k-1)} + \omega b, \tag{20.14}$$

leading us to the specification of the SOR matrix.

$$x^{(k)} = B_{\text{SOR}} x^{(k-1)} + c_{\text{SOR}}$$

where

$$B_{\text{SOR}} = (D + \omega L)^{-1} ((1 - \omega)\, D - \omega U), \quad c_{\text{SOR}} = \omega\, (D + \omega L)^{-1} b \tag{20.15}$$

20.4.4 Conditions Guaranteeing Convergence

You may try an initial value, say $x^{(0)} = 0$ and find that your chosen iterative method diverges. Thus, it is helpful to be aware of conditions you know will guarantee convergence for any initial approximation. These conditions are dependent on properties of B in the matrix formulation of the iteration, particularly, the norms and the spectral radius of B. The following theorem provides a simple test for convergence of any iteration with matrix form $x^{(k)} = Bx^{(k-1)} + c$.

Theorem 20.1. *If the matrix B in the iteration $x^{(k)} = Bx^{(k-1)} + c$ has the property that $\|B\| < 1$ for some subordinate norm, then the iteration converges for any choice of $x^{(0)}$.*

Proof. Assume that $\|B\| < 1$ for some subordinate norm. Then $\left\|B^k\right\| \leq \|B\|^k \to 0$ as $k \to \infty$, and so $B^k \to 0$ as $k \to \infty$. Now consider the iteration

$$x^{(k)} = Bx^{(k-1)} + c. \tag{20.16}$$

If x is the true solution,

$$x = Bx + c, \tag{20.17}$$

by subtracting Equation 20.16 from Equation 20.17, we get $x - x^{(k)} = (Bx + c) - \left(Bx^{(k-1)} + c\right)$, and

$$x - x^{(k)} = B(x - x^{(k-1)}). \tag{20.18}$$

If we let $e^{(k)} = x - x^{(k)}$ be the error at the kth step of the iteration, then Equation 20.18 gives the relation

$$e^{(k)} = Be^{(k-1)}. \tag{20.19}$$

Applying Equation 20.18 repeatedly, $e^{(1)} = Be^{(0)}$, $e^{(2)} = Be^{(1)} = B^2e^{(0)}$, $e^{(3)} = Be^{(2)} = B^3e^{(0)}$, and by induction $e^{(k)} = B^ke^{(0)}$. Since $B^k \to 0$ as $k \to \infty$, it follows that $\lim_{k\to\infty} e^{(k)} = 0$, and the iteration converges. □

Example 20.4. For the matrix in Examples 20.1 and 20.2

$$D = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & -7 \end{bmatrix}, \quad L = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 1 & 6 & 0 \end{bmatrix}, \quad U = \begin{bmatrix} 0 & -1 & 2 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Using $\omega = 1.2$ for SOR, Table 20.1 lists matrix B for each method along with its subordinate norms. In each case, one of the norms is less than 1, so convergence is guaranteed. Note that for the SOR iteration, $\|B\|_\infty = 0.9255$. You might suspect that the iteration converges slowly.

TABLE 20.1 Convergence of Iterative Methods

Method	B	$\|\|B\|\|_1$	$\|\|B\|\|_\infty$	$\|\|B\|\|_2$
Jacobi	$\begin{bmatrix} 0 & 0.2000 & -0.4000 \\ 0.2500 & 0 & -0.2500 \\ 0.1429 & 0.8571 & 0 \end{bmatrix}$	1.0571	1.0000	0.8997
Gauss-Seidel	$\begin{bmatrix} 0 & 0.2000 & -0.4000 \\ 0 & 0.0500 & -0.3500 \\ 0 & 0.0714 & -0.3571 \end{bmatrix}$	1.1071	0.6000	0.6692
SOR	$\begin{bmatrix} -0.2000 & 0.2400 & -0.4800 \\ -0.0600 & -0.1280 & -0.4440 \\ -0.0690 & -0.0905 & -0.7390 \end{bmatrix}$	1.6630	0.9255	1.0063

■

The condition $\|B\| < 1$ for some subordinate matrix norm guarantees convergence. However, it is possible for the iteration to converge even if $\|B\| \geq 1$ for the Jacobi and Gauss-Seidel iterations (Problem 20.6). As a first check to determine if an iteration converges, compute the subordinate norms of B and determine if one has value less than 1. If so, you can proceed; however, if the subordinate norms are all greater than or equal to 1, the iteration may still converge. Recall that the spectral radius of an $n \times n$ square matrix B, $\rho(B)$, is defined by $\rho(B) = \max_{1\leq i\leq n} |\lambda_i|$, where λ_i are the eigenvalues of B. Find the spectral radius, $\rho(B)$, and verify that $\rho(B) < 1$; in other words, $\rho(B) < 1$ guarantees convergence. It is also the case that if an iteration converges, then $\rho(B) < 1$, so if $\rho(B) \geq 1$, the iterative method will not converge.

Theorem 20.2. *The iteration $x^{(k+1)} = Bx^{(k)} + c$ converges if and only if $\rho(B) < 1$.*

Proof. Assume that the iteration converges. If $e^{(k)} = x^{(k)} - x$, as in the proof of Theorem 20.1, $e^{(k)} = B^ke_0$. Since $\lim_{k\to\infty} e^{(k)} = 0$, it follows that $\lim_{k\to\infty} B^k \to 0$. As a result, $\lim_{k\to\infty} \left\|B^kx\right\|_2 \leq \lim_{k\to\infty} \left\|B^k\right\|_2 \|x\|_2$, and $\lim_{k\to\infty} B^kx = 0$ for all vectors $x \in \mathbb{R}^n$. If we assume that $\rho(B) \geq 1$, there must be an eigenvector u corresponding to eigenvalue λ with

$|\lambda| \geq 1$. Since $Bu = \lambda u, B^2u = \lambda^2 u, B^3u = \lambda^3 u, \ldots, B^k u = \lambda^k u$, and it is not true that $\lim_{k\to\infty} B^k u = 0$. By contradiction, $\rho(B) < 1$.

Proving that if $\rho(B) < 1$, then $x^{(k+1)} = Bx^{(k)} + c$ converges will be omitted. For a proof, see Refs. [1, p. 280], [2, p. 614], and [27, pp. 143-145]. □

Example 20.5. In Example 20.3, the Jacobi iteration failed, but the Gauss-Seidel and SOR methods succeeded. The spectral radius of each iteration matrix is

$$\begin{aligned} \rho(B_{\text{GS}}) &= 0.5 \\ \rho(B_{\text{SOR}}) &= 0.3687 \\ \rho(B_{\text{J}}) &= 1.1372, \end{aligned}$$

so these are the results expected. ■

Example 20.6. For the matrix $A = \begin{bmatrix} 1 & 4 & -1 \\ 2 & -1 & 5 \\ 1 & 0 & 3 \end{bmatrix}$, $B_{\text{GS}} = \begin{bmatrix} 0.0000 & -4.0000 & 1.0000 \\ 0.0000 & -8.0000 & 7.0000 \\ 0.0000 & 1.3333 & -0.3333 \end{bmatrix}$, and $\rho(B_{\text{GS}}) = 9.0685$. The Gauss-Seidel iteration will not converge. The same is true for the Jacobi iteration. Verify that $\rho(B_{\text{J}}) = 2.9825$. ■

20.4.5 The Spectral Radius and Rate of Convergence

Intuitively, there should be a link between the spectral radius of the iteration matrix B and the rate of convergence. Suppose that B has n linearly independent eigenvectors, $v_1, v_2, \ldots, v_n$ and associated eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$. Use the notation of Theorems 20.1 and 20.2 for the error $e^{(k)}$. Since the eigenvectors are a basis,

$$e^{(0)} = \sum_{i=1}^{n} c_i v_i.$$

It follows that:

$$\begin{aligned} e^{(1)} &= Be^{(0)} = \sum_{i=1}^{n} c_i B v_i = \sum_{i=1}^{n} c_i \lambda_i v_i \\ e^{(2)} &= Be^{(1)} = \sum_{i=1}^{n} c_i \lambda_i B v_i = \sum_{i=1}^{n} c_i \lambda_i^2 v_i. \end{aligned}$$

By continuing in this fashion, there results

$$e^{(k)} = \sum_{i=1}^{n} c_i \lambda_i^k v_i.$$

Let $\rho(B) = \lambda_1$ and suppose that $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq \lambda_n$ so that

$$\begin{aligned} e^{(k)} &= c_1 \lambda_1^k v_1 + \sum_{i=2}^{n} c_i \lambda_i^k v_i \\ &= \lambda_1^k \left(c_1 v_1 + \sum_{i=2}^{n} c_i \left(\frac{\lambda_i}{\lambda_1} \right)^k v_i \right). \end{aligned}$$

As k becomes large, $\left(\frac{\lambda_i}{\lambda_1}\right)^k$, $2 \leq i \leq n$ becomes small and we have

$$e^{(k)} \approx \lambda_1^k c_1 v_1.$$

This says that the error varies with the kth power of the spectral radius and that the spectral radius is a good indicator for the rate of convergence.

20.4.6 Convergence of the Jacobi and Gauss-Seidel Methods for Diagonally Dominant Matrices

A matrix is *strictly row diagonally dominant* if the absolute value of the diagonal element is greater than the sum of the absolute values of the off-diagonal elements in its row.

$$|a_{ii}| > \sum_{j=1, j\neq i}^{n} |a_{ij}|, \quad i = 1, 2, \ldots, n \tag{20.20}$$

We will prove that when A is strictly row diagonally dominant, the Jacobi iteration will converge. The reverse is not true. There are matrices that are not strictly row diagonally dominant for which the iteration converges. The matrix of Examples 21.1 and 21.2 is an example.

Theorem 20.3. *If A is strictly row diagonally dominant, then the Jacobi iteration converges for any choice of the initial approximation $x^{(0)}$.*

Proof. Recall that the matrix for the Jacobi iterative method is $B_J = -D^{-1}(L+U)$. Then

$$B_J = -D^{-1}(L+U) = \begin{bmatrix} -\frac{1}{a_{11}} & & & 0 \\ & -\frac{1}{a_{22}} & & \\ & & \ddots & \\ 0 & & & -\frac{1}{a_{nn}} \end{bmatrix} \begin{bmatrix} 0 & a_{12} & \ldots & a_{1n} \\ a_{21} & 0 & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & \ldots & \ldots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & -\frac{a_{23}}{a_{22}} & \ldots & -\frac{a_{2n}}{a_{22}} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & -\frac{a_{n-1,n}}{a_{n-1,n-1}} \\ -\frac{a_{n1}}{a_{nn}} & \ldots & \ldots & -\frac{a_{n,n-1}}{a_{nn}} & 0 \end{bmatrix}$$

Recall that for any square matrix G,

$$\|G\|_\infty = \max_{1\leq i\leq n} \sum_{j=1}^{n} |g_{ij}|.$$

For any row i of B_J,

$$\sum_{j=1, j\neq i}^{n} \left| -\frac{a_{ij}}{a_{ii}} \right| = \frac{1}{|a_{ii}|} \sum_{j=1, j\neq i}^{n} |a_{ij}| < 1$$

by Equation 20.20. Thus, $\|B_J\|_\infty < 1$, and by Theorem 20.1, the Jacobi method converges. □

If A is strictly row diagonally dominant, the Gauss-Seidel iteration converges for any choice of the initial approximation $x^{(0)}$. For a proof, see Ref. [1, pp. 286-288]. In Ref. [2, pp. 615-616], you will find the proof of another result concerning the Gauss-Seidel method. It is particularly useful, since many engineering problems involve symmetric positive definite matrices.

Theorem 20.4. *Let A be a symmetric positive definite matrix. For any arbitrary choice of initial approximation $x^{(0)}$, the Gauss-Seidel method converges.*

It should be noted that for matrices that are not row diagonally dominant, there are examples where the Jacobi iteration converges and the Gauss-Seidel iteration diverges. Similarly, there are examples of matrices for which the Gauss-Seidel method converges and the Jacobi method diverges (see Ref. [11]).

20.4.7 Choosing ω for SOR

We have seen that poor choice of the relaxation parameter ω can lead to poor convergence rates. On the other hand, a good choice of ω can lead to very fast convergence compared to the Jacobi or Gauss-Seidel methods. The spectral radius of B_{SOR} is the eigenvalue of B_{SOR} with maximum magnitude, and we want to choose ω so that $|\rho(B_{\text{SOR}}(\omega))|$ is a minimum. Finding the optimal value of ω is very difficult in general, and the optimal value is known only for special types of matrices. It is known, however, that ω must satisfy $0 < \omega < 2$ [1, p. 290].

Theorem 20.5. *If the SOR iteration converges for every initial approximation $x^{(0)}$, then $0 < \omega < 2$.*

This result says that you never choose a relaxation parameter ω outside the range $(0, 2)$. Normally, the choice is overrelaxation ($\omega > 1$) to put the greatest weight on the newly computed values. If $\|B_{\text{SOR}}\| < 1$ for some subordinate norm or $\rho(B_{\text{SOR}}) < 1$, the SOR iteration converges. The following theorem [1, p. 290], which we state without proof, provides another criterion that guarantees convergence.

Theorem 20.6. *If A is a symmetric positive definite matrix and $0 < \omega < 2$, the SOR and Gauss-Seidel iterations converge for any choice of $x^{(0)}$.*

Example 20.7. Let A be the 10×10 pentadiagonal matrix

$$A = \begin{bmatrix} 6 & -2 & -1 & & & & & & & \\ -2 & 6 & -2 & -1 & & & & & & \\ -1 & -2 & 6 & \ddots & \ddots & & & & & \\ & -1 & \ddots & \ddots & \ddots & -1 & & & & \\ & & \ddots & -2 & 6 & -2 & & & & \\ & & & -1 & -2 & 6 & & & & \end{bmatrix}$$

that can be built using the function `pentd` in the software distribution. The function `optomega` in the book software distribution approximates the optimal ω for a matrix and graphs $\rho(B_{\text{SOR}}(\omega))$ as a function of ω, $0 < \omega < 2$. It should be noted that the function `optomega` is provided for demonstration purposes only, and is not intended for use with a large matrix. The following code estimates the optimal ω for A and then applies the SOR iteration to solve the system $Ax = b$, where b is $\begin{bmatrix} 1 & 1 & \dots & 1 & 1 \end{bmatrix}^{\text{T}}$. It prints the number of iterations required to attain a error tolerance of 1.0×10^{-14} and the relative residual. Figure 20.1 is the graph produced by `optomega`.

```
>> woptimal = optomega(A)

woptimal =
    1.4600

>> [x,iter,relresid] = sor(A,rhs,x0,woptimal,1.0e-14,100);
>> iter

iter =
    54

>> relresid

ans =
  9.2334e-015
```

Try another value of ω such as 1.3 and observe a slower convergence rate. ■

20.5 APPLICATION: POISSON'S EQUATION

Poisson's equation is one of the most important equations in applied mathematics and has applications in such fields as astronomy, heat flow, fluid dynamics, and electromagnetism. Let R be a bounded region in the plane with boundary ∂R (Figure 20.2), $g(x, y)$ be defined on ∂R, and $f(x, y)$ be a function defined in R. Find a function $u(x, y)$ such that

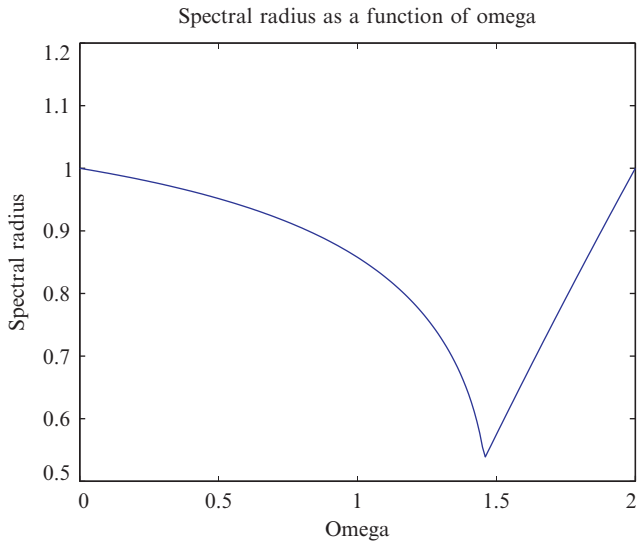

FIGURE 20.1 SOR spectral radius.

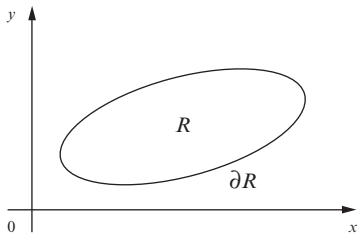

FIGURE 20.2 Region in the plane.

$$-\frac{\partial^2 u}{\partial x^2}-\frac{\partial^2 u}{\partial y^2}=f\,(x,\,y)\,,$$

$$u\,(x,\,y)=g\,(x,\,y)\quad \text{on } \partial R$$

Rarely is an analytical solution known, so approximation techniques must be used. We will use the finite difference method to obtain a numerical solution and assume for simplicity that the region R is the unit square $0 \le x \le 1,\ 0 \le y \le 1$. Divide the square into a grid of small squares having sides of length $h = \frac{1}{n}$ (Figure 20.3). In Section 12.2, we studied the heat equation in one spacial variable and approximated the second partial derivative by a difference equation

$$\frac{\partial^2 u}{\partial x^2}\left(x_i,\,t_j\right)\approx\frac{1}{h^2}\left(u_{i-1,j}-2u_{i,j}+u_{i+1,j}\right),$$

where h is the equal spacing between points. We will do the same thing here for the two partial derivatives to obtain the difference approximation

$$\frac{1}{h^2}\left(-u_{i-1,j}+2u_{i,j}-u_{i+1,j}\right)+\frac{1}{h^2}\left(-u_{i,j-1}+2u_{i,j}-u_{i,j+1}\right)=f\left(x_i,\,y_j\right)$$

If we multiply by h^2 and collect terms, the result is the set of equations

$$-u_{i-1,j}-u_{i+1,j}+4u_{ij}-u_{i,j-1}-u_{i,j+1}=h^2 f\left(x_i,\,y_j\right),\quad 1\le i,\,j\le n-1 \tag{20.21}$$

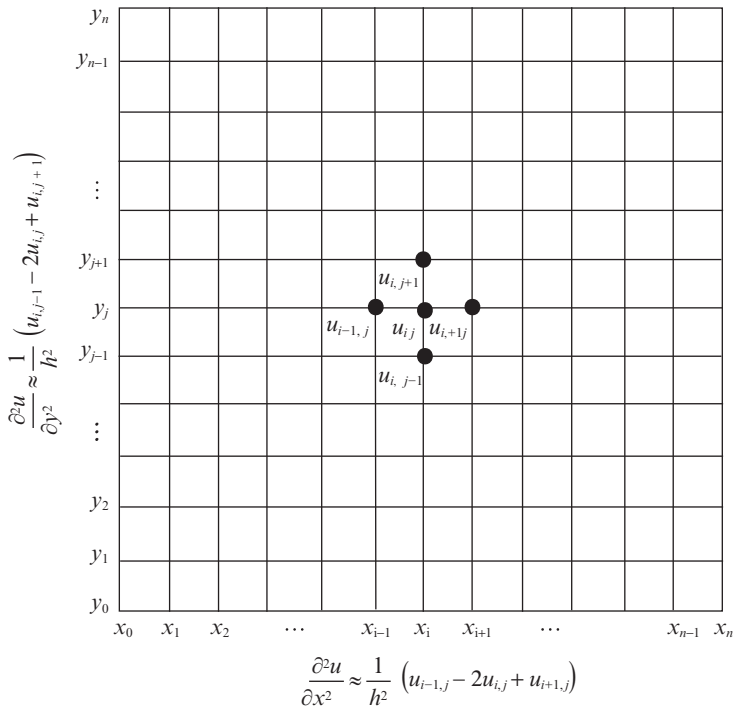

FIGURE 20.3 Five-point stencil.

Each equation contains five values at grid points and forms what is called a *five-point central difference approximation*, also called a *five-point stencil* (Figure 20.3).

When one or two of the four points around the center touch the boundary, the boundary condition must be used. For instance, if $i = 1$, then the equation centered at $(1, 1)$ is

$$-u_{0,1} - u_{2,1} + 4u_{11} - u_{1,0} - u_{1,2} = h^2 f(x_1, y_1)$$

or

$$-u_{2,1} + 4u_{11} - u_{1,2} = h^2 f(x_1, y_1) + g(0, y_1) + g(x_1, 0) \tag{20.22}$$

Execute the SOR iteration by solving for u_{ij} using Equation 20.21 to obtain

$$u_{ij} = \omega\left(\frac{u_{i-1,j} + u_{i+1,j} + u_{i,j-1} + u_{i,j+1} + h^2 f(x_i, y_i)}{4}\right) + (1 - \omega)\, u_{ij}. \tag{20.23}$$

In Ref. [23, p. 564], it is stated that the optimal value of ω is

$$\omega_{\text{opt}} = \frac{2}{1 + \sin \pi h}. \tag{20.24}$$

We have not specified the matrix form of the finite difference approximation, but will do so in Section 21.11. The matrix is symmetric and positive definite, so we know the SOR iteration will converge by Theorem 20.6. The function `sorpoisson` in the software distribution assigns the boundary values, assigns values of $f(x, y)$ in the interior, and executes the SOR iteration.

```
%SORPOISSON Numerically approximates the solution of the Poisson
%equation on the square 0 <= x,y <= 1
%
%   [x y u] = sorpoisson(n,f,g,omega,numiter) computes the solution.
%   n is the number of subintervals, f is the right-hand side, g is
%   the boundary condition on the square, omega is the relaxation parameter,
%   and numiter is the number of SOR iterations to execute. x and y are
%   the grid of points on the x and y axes, and u is the matrix containing
```

```
%   the numerical solution. Upon building x, y, and u, sorpoisson draws a
%   surface plot of the solution.
%
```

Example 20.8. Consider the Poisson equation

$$\begin{aligned} -\frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial y^2} &= 20\pi^2 \sin(2\pi x)\sin(4\pi y) \\ u(x, y) &= \sin(2\pi x)\sin(4\pi y) \text{ on } \partial R \end{aligned}$$

whose exact solution is $u(x, y) = \sin(2\pi x)\sin(4\pi y)$. Apply the SOR iteration 20.23 with $n = 100$ and numiter $= 125$. Using Equation 20.24, assign $\omega = 1.9196$. Figure 20.4 is a graph of the solution obtained from the SOR iteration, and Figure 20.5 shows the actual solution. ■

Problems 20.9–20.14 deal with the Jacobi and Gauss-Seidel iterations for the one-dimensional Poisson equation. These problems deal with the coefficient matrix and its relation to the convergence of the iterations.

20.6 CHAPTER SUMMARY

The Jacobi Iteration

The Jacobi iteration is the simplest of the classical iterative methods and, generally, the slowest. However, it forms a basis for the understanding of other methods, such as Gauss-Seidel and SOR. Starting with an initial approximation, $x^{(0)}$, to the solution of $Ax = b$, use the first equation to compute $x_1^{(1)}$ in terms of $x_2^{(0)}, x_3^{(0)}, \ldots, x_n^{(0)}$. Now use the second equation to compute $x_2^{(1)}$ in terms of $x_1^{(0)}, x_3^{(0)}, \ldots, x_n^{(0)}$. In general, the equation for the computation is

$$x_i^{(k)} = \frac{1}{a_{ii}}\left[b_i - \left(\sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k-1)}\right)\right], \quad 1 \leq i \leq n.$$

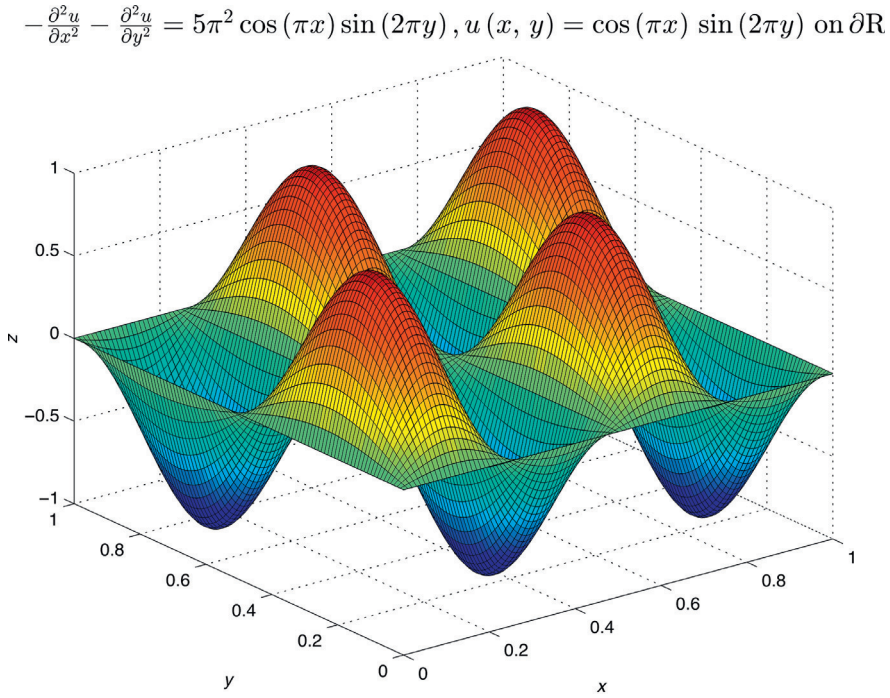

FIGURE 20.4 Poisson's equation. (a) Approximate solution and (b) analytical solution.

Continued

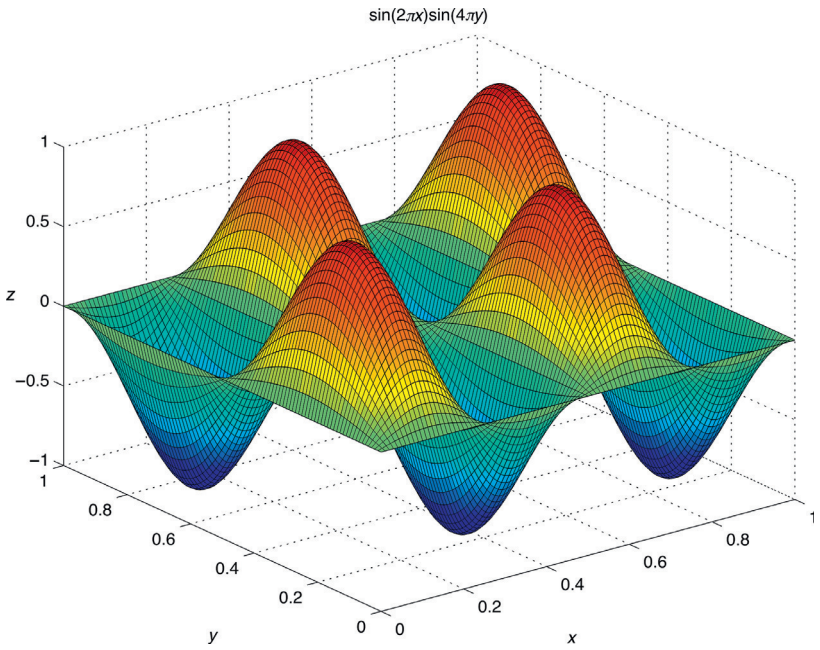

FIGURE 20.4 , CONT'D

The Jacobi method does not make use of new components of the approximate solution as they are computed. This requires storing both the previous and the current approximations. If A is strictly row diagonally dominant, then the Jacobi iteration converges for any choice of the initial approximation $x^{(0)}$. However, the Jacobi iteration may converge for a matrix that is not strictly row diagonally dominant.

The Gauss-Seidel Iteration

In general, the Gauss-Seidel iteration is an improvement over the Jacobi iteration because it uses the approximation to a component of the solution as soon as it is available. For instance, after computing $x_1^{(1)}$, it is used in the computation of $x_2^{(1)}$. The general formula for the iteration is

$$
\begin{aligned}
x_1^{(k)} &= \frac{1}{a_{11}}\left[b_1 - \left(\sum_{j=2}^{n} a_{1j}x_j^{(k-1)}\right)\right] \\
x_i^{(k)} &= \frac{1}{a_{ii}}\left[b_i - \left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}\right)\right], \quad i = 2, 3, \ldots, n-1 \\
x_n^{(k)} &= \frac{1}{a_{nn}}\left[b_n - \sum_{j=1}^{n-1} a_{nj}x_j^{(k)}\right]
\end{aligned}
$$

Unlike the Jacobi method, this method requires storing only one vector rather than two. The Gauss-Seidel iteration is guaranteed to converge for any initial approximation if A is strictly diagonally dominant and when A is symmetric and positive definite.

The SOR Iteration

This iteration computes $x_i^{(k)}$ by forming a weighted average between the previous value $x_i^{(k-1)}$ and the value computed by the Gauss-Seidel iteration, and the formula is

$$x_i^{(k)} = \frac{\omega}{a_{ii}}\left[b_i - \left(\sum_{j=1}^{i-1} a_{ij}x_j^{(k)} + \sum_{j=i+1}^{n} a_{ij}x_j^{(k-1)}\right)\right] + (1-\omega)\,x_i^{(k-1)}, \quad i = 1, 2, \ldots, n.$$

The idea is to choose a value of ω that will accelerate the rate of convergence of the iteration. It is known that unless $0 < \omega < 2$, the method will not converge. There is no general formula for an optimal ω. Usually $\omega > 1$, but $\omega < 1$ (underrelaxation) sometimes gives better results. SOR converges for any ω if A is positive definite, and positive definite matrices appear in many engineering problems.

Convergence of the Basic Iterative Methods

Analysis of convergence criteria for the basic iterations is based upon writing the equations defining the iteration in the matrix form

$$x^{(k)} = Bx^{(k-1)} + c. \tag{20.25}$$

If $\|B\| < 1$ for any subordinate norm, then any iteration 20.25 converges. The matrices B_J, B_{GS}, and B_{SOR} for the Jacobi, Gauss-Seidel, and SOR methods, respectively, are determined in Sections 20.4.1–20.4.3. Note that this is a sufficient condition only. It is possible for the iteration 20.25 to converge if $\|B\| \geq 1$. A necessary and sufficient condition for an iteration to converge is that the spectral radius of B ($\rho(B)$) is less than 1. While this result is of significant theoretical importance, computing the spectral radius is computationally costly. It is appropriate to first check to see if $\|B\| < 1$. Also, the Jacobi and Gauss-Seidel methods converge if A is strictly diagonally dominant, and the Gauss-Seidel iteration converges if A is positive definite. Convergence of the SOR iteration is guaranteed if $0 < \omega < 2$ and A is positive definite.

The Poisson Equation

Poisson's equation is very important in many areas of application. Section 20.5 develops a five-point central difference approximation for the two-dimensional Poisson equation. The coefficient matrix is positive definite, so the SOR iteration applies. This is one of the rare cases where the optimal choice for ω is known, and a function `sorpoisson` in the book software distribution uses the SOR iteration to numerically approximate and make a surface plot of the solution to the Poisson equation on the unit square $0 \leq x, y \leq 1$.

20.7 PROBLEMS

20.1 Perform three iterations of the Jacobi method using pencil and paper. Use $x^{(0)} = 0$.

$$\begin{bmatrix} 3 & 1 \\ 2 & 5 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

20.2 Do Problem 21.2 using the Gauss-Seidel iteration.

20.3 Do Problem 21.1 using SOR with $\omega = 1.2$.

20.4 Let $A = \begin{bmatrix} \frac{1}{6} & 0 & \frac{1}{8} \\ 0 & \frac{1}{3} & \frac{1}{7} \\ \frac{1}{5} & 0 & \frac{1}{4} \end{bmatrix}$. What can you say about the convergence of the Jacobi and Gauss-Seidel iterations?

20.5 Let A be the general 2×2 matrix $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$.

a. Compute exact formulas for B_J defined by Equation 20.8 and B_{GS} defined by Equation 20.11. Consider using the MATLAB Symbolic Math Toolbox.

b. Determine analytic formulas for the eigenvalues of B_J and B_{GS}.

c. Using the result of (b), determine if either or both of the Jacobi and Gauss-Seidel iterations converge for $A = \begin{bmatrix} 3 & -4 \\ 4 & 5 \end{bmatrix}$.

d. Answer part (c) for the matrix $A = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}$.

e. Is it possible for one of Jacobi and Gauss-Seidel to converge and the other diverge for a 2×2 matrix?

20.6 Let $A = \begin{bmatrix} 2 & 2 \\ 1 & 3 \end{bmatrix}$.

a. Show that the Jacobi and Gauss-Seidel iterations converge, but that the iteration matrices B_{J} and B_{GS} have one-, two-, and infinity norms greater than or equal to 1.

b. The matrix

$$B_{\mathrm{SOR}}(\omega) = \begin{bmatrix} 1-\omega & -\omega \\ \frac{\omega(2\omega-2)}{6} & \frac{\omega^2}{3} - \omega + 1 \end{bmatrix}$$

Let ω range from 0.01 in steps of 0.01-1.99, and graph ω vs. $\rho(B_{\mathrm{SOR}}(\omega))$.

c. Do part (b), except graph ω vs. $\|B_{\mathrm{SOR}}(\omega)\|_2$.

d. Comment on the results of parts (a)-(c).

20.7 Show that

a. The Jacobi iteration converges for $A_1 = \begin{bmatrix} 2 & 1 & -2 \\ 1 & 1 & 1 \\ 3 & 2 & 1 \end{bmatrix}$ but the Gauss-Seidel iteration does not converge.

b. The Gauss-Seidel iterations converges for $A_2 = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 2 & 1 \\ 1 & 1 & 3 \end{bmatrix}$ but the Jacobi iteration does not.

20.8 If $\|\cdot\|$ is a subordinate matrix norm, prove that

$$\rho(A) < \|A\|.$$

Assume the result of Theorem 20.2, and provide an alternate proof of Theorem 20.1.

Problems 20.9–20.14 deal with the one-dimensional Poisson equation,

$$-\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = f(x), \quad 0 \le x \le 1, \; u(0) = u(1) = 0 \tag{20.26}$$

In most cases, it is not possible to find an analytical solution, so an approximation technique must be used. Divide the interval into n subintervals of width $h = \frac{1}{n}$, giving $n+1$ points

$$\{x_1 = 0, x_2 = h, \ldots, x_i = (i-1)h, \ldots, x_n = (n-1)h, x_{n+1} = 1\}.$$

20.9 Using Taylor series, it can be shown that $\frac{\mathrm{d}^2 u}{\mathrm{d}x^2}(x_i) \approx \frac{u(x_{i+1}) - 2u(x_i) + u(x_{i-1})}{h^2}$. Use the notation $u_i = u(x_i)$ and show that after using the approximation for the second derivative in the differential equation, the result is the system of equations (Figure 20.5)

$$-u_{i+1} + 2u_i - u_{i-1} = h^2 f(x_i), \quad 2 \le i \le n.$$

20.10

a. Noting that $u_1 = 0, u_{n+1} = 0$, show that the system of equations in Problem 21.11 can be written in matrix form $Ax = b$, where

$$A = \frac{1}{h^2}\begin{bmatrix} 2 & -1 & 0 & \ldots & 0 \\ -1 & 2 & -1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & -1 & 2 & -1 \\ 0 & & & -1 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} f(x_2) \\ f(x_3) \\ \vdots \\ f(x_{n-1}) \\ f(x_n) \end{bmatrix}.$$

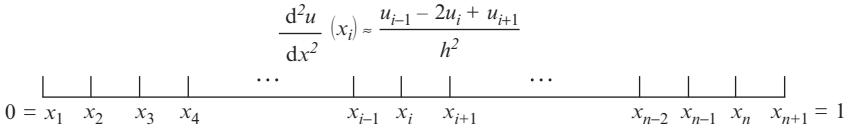

FIGURE 20.5 One-dimensional Poisson equation grid.

b. Show that A is positive definite by proving that $x^{\mathrm{T}}Ax > 0$ for all $x \neq 0$. Hint: Show that $x^{\mathrm{T}}Ax = \langle Ax, x\rangle$. Multiply this out and look for terms of the form $(x_{i+1} - x_i)^2$.

20.11 In this problem, you will find the eigenvalues of matrix A in Problem 20.10.

a. Matrix A has dimension $(n-1) \times (n-1)$. Let $v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_{n-2} \\ v_{n-1} \end{bmatrix}$ be an eigenvector of A. Letting $h = \frac{1}{n}$, show that

$$-v_{i-1} + \left(2 - \lambda h^2\right) v_i - v_{i+1} = 0, \quad 1 \leq i \leq n-1,$$

where $v_0 = v_n = 0$.

b. Try solutions of the form $v_j = \sin(\omega j)$, $0 \leq j \leq n$ and show that

$$\lambda_j = \frac{4}{h^2} \sin^2\left(\frac{\pi j}{2n}\right), \quad 1 \leq j \leq n-1$$

are distinct eigenvalues of A. Hint: Use trigonometric identities including $1 - \cos(\omega) = 2\sin^2\left(\frac{\omega}{2}\right)$.

c. We know that A is positive definite. Are its eigenvalues consistent with that?

d. Find the condition number of A, and show that the matrix is ill-conditioned for large n.

Remark 20.1. With homogeneous boundary conditions $u(0) = u(1) = 0$, it can be shown due to the form of the right-hand side b that

$$\frac{\left\|\delta u^{(n)}\right\|_2}{\left\|u^{(n)}\right\|_2} \leq C \frac{\left\|\delta b^{(n)}\right\|_2}{\left\|b^{(n)}\right\|_2},$$

where C is a constant independent of n. In short, the system $Ax = b$ can be solved reliably (see Ref. [27, pp. 87-88]).

20.12 To solve the problem $Ax = b$ using the Jacobi iteration, the spectral radius of the matrix B_{J} must be less than one.

a. Using the result of Problem 20.11(b), show that the eigenvalues of B_{J} are

$$\mu_j = 1 - \frac{\lambda_j}{2n^2},$$

where λ_j are the eigenvalues of A. Hint: Show that $B_{\mathrm{J}} = I - D^{-1}A$, and let v be an eigenvector of B_{J} with corresponding eigenvalue μ.

b. Show that the spectral radius of B_{J} is

$$\rho(B_{\mathrm{J}}) = \cos\left(\frac{\pi}{n}\right),$$

and that $\rho(B_{\mathrm{J}}) < 1$. Thus, the Jacobi method converges. Hint: Use the half-angle formula $\sin^2\left(\frac{\theta}{2}\right) = \frac{1}{2}(1 - \cos\theta)$.

c. Using the McLaurin series for $\cos\frac{\pi}{n}$, show that

$$\rho(B_{\mathrm{J}}) = 1 - \frac{\pi^2}{2n^2} + O\left(h^4\right).$$

20.13

a. It can be shown that the spectral radius of the iteration matrix B_{GS} for the Gauss-Seidel iteration is $\rho(B_{\mathrm{GS}}) = \cos^2\left(\frac{\pi}{n}\right) < 1$ [27, p. 155]. Show that

$$\rho(B_{\mathrm{GS}}) = 1 - \frac{\pi^2}{n^2} + O\left(h^4\right).$$

b. Explain why for large values of n, the Gauss-Seidel method converges faster.

20.14 There is a means of handling the Gauss-Seidel iteration for the one-dimensional Poisson equation that involves coloring half the unknown values red and half black. This method allows parallelism, whereas the iteration 20.4

FIGURE 20.6 One-dimensional red-black GS.

does not, and the modification leads to best case convergence results that depend on it [1, pp. 282-283, 285-294]. Color x_2 light gray (represents red), x_3 black, and so forth, as shown in Figure 20.6.

a. If a grid point is red, what are the colors of its neighbors? If a grid point is black, what is the color of its neighbors?

b. To apply Gauss-Seidel, begin with x_2 and apply the difference formula to all red points. Now start at x_3 and apply the difference formula to all the black points. Sketch an algorithm that uses this coloring to implement the Gauss-Seidel iteration.

c. Explain why this ordering allows the iteration to be parallelized.

20.15 In finding a numerical solution to the two-dimensional Poisson equation, we used row ordering to define the finite difference equations. There is a better way, called *red-black ordering*. Think of the grid as a chessboard. Begin in the lower left corner of the grid of unknowns and color (1, 1) red. Now alter red and black until you have colored $(n-1, n-1)$.

a. Draw the board corresponding to $n = 7$, and color the interior grid points.

b. In the five-point central difference scheme, if (i, j) is red, what color are its four neighbors? What color are the neighbors of a black grid point?

c. In terms of indices i, j, when is u_{ij} red, and when is it black?

d. To apply Gauss-Seidel, begin with red grid point (1, 1) and apply the difference formula to all the red points. Now start with the black grid point (2, 1) and apply the difference formula to the black points. Sketch an algorithm that implements this version of Gauss-Seidel.

e. Explain why this ordering allows the iteration to be parallelized.

20.7.1 MATLAB Problems

20.16 Using the Jacobi and Gauss-Seidel methods, solve the system with tol $= 0.5 \times 10^{-14}$ and numiter $= 50$. In each case, print the number of iterations required and the relative residual $\|b - Ax\|_2 / b$.

$$\begin{bmatrix} 6 & -1 & 2 & 1 \\ 1 & 6 & 1 & -1 \\ 0 & 1 & 3 & 1 \\ 1 & -2 & 1 & 5 \end{bmatrix} \begin{bmatrix} 3 \\ 2 \\ -6 \\ 1 \end{bmatrix}.$$

20.17 This problem investigates convergence properties. The matrix $A = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{bmatrix}$ is positive definite (Problem 20.10(b)).

a. Is matrix A strictly row diagonally dominant?

b. Is the Jacobi method guaranteed to converge? What about the Gauss-Seidel and SOR iterations?

c. If the Gauss-Seidel method converges, solve $Ax = \begin{bmatrix} 1 \\ 3 \\ 7 \end{bmatrix}$.

d. If the SOR iteration converges, use `optomega` to estimate an optimal ω, and demonstrate that `sor` improves the convergence rate relative to Gauss-Seidel using tol $= 1.0 \times 10^{-10}$ and maxiter = 100.

20.18 Let $A = \begin{bmatrix} 0.1000 & 0.5000 & -0.1000 \\ 0.4000 & 0.2000 & 0.6000 \\ 0.2000 & -0.3000 & 0.4000 \end{bmatrix}$. The matrix A is not diagonally dominant.

a. Compute $\|B_J\|_1$, $\|B_J\|_\infty$, $\|B_J\|_2$ and $\|B_{GS}\|_1$, $\|B_{GS}\|_\infty$, $\|B_{GS}\|_2$.

b. Compute the spectral radius of B_J and B_{GS}.

c. Does either method converge? Demonstrate your answer by using a random b and $x_0 = 0$.

20.19 In Chapter 12, we used finite difference methods to approximate the solution of the heat equation in one spacial variable. We needed to solve a tridiagonal linear system with coefficient matrix

$$B = \begin{bmatrix} 1+2r & -r & 0 & \cdots & 0 \\ -r & 1+2r & -r & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \vdots & -r & 1+2r & -r \\ 0 & 0 & 0 & -r & 1+2r \end{bmatrix}.$$

This matrix is strictly row diagonally dominant, and so both the Jacobi and Gauss-Seidel iterations will converge.

a. Let $r = 0.25$ and solve the 100×100 linear system $Bu = c$, with $c = \begin{bmatrix} 1 & 1 & \ldots & 1 & 1 \end{bmatrix}^{\mathrm{T}}$ using the Jacobi iteration with tol $= 0.5 \times 10^{-14}$, numiter $= 100$, and $x_0 = 0$.

b. Do part (a) using the Gauss-Seidel iteration.

c. Do part (a) using SOR with $\omega = \{1.1, 1.2, 1.3, 1.5, 1.9\}$. Which value of ω works best?

d. Solve the system using the MATLAB function `thomas` introduced in Section 9.4.

e. Solve the system using the MATLAB command "`B\c`."

f. Compare the results of parts (a)-(e).

20.20

a. Show that the matrix

$$A = \begin{bmatrix} 4 & -2 & 0 & 0 & 0 & 0 & 0 \\ -2 & 4 & -2 & 0 & 0 & 0 & 0 \\ 0 & -2 & 4 & -2 & 0 & 0 & 0 \\ 0 & 0 & -2 & 4 & -2 & 0 & 0 \\ 0 & 0 & 0 & -2 & 4 & -2 & 0 \\ 0 & 0 & 0 & 0 & -2 & 4 & -2 \\ 0 & 0 & 0 & 0 & 0 & -2 & 4 \end{bmatrix}$$

is positive definite.

b. Compute $\rho(B_{\mathrm{J}})$, $\rho(B_{\mathrm{GS}})$, and $\rho(B_{\mathrm{SOR}}(1.2))$. Project a ranking for the number of iterations required by the methods.

c. Generate a random 7×1 vector *rhs* and solve $Ax = rhs$ using all three methods. Are your projections in part (b) correct?

d. For the matrix $A = \begin{bmatrix} -1 & 3 & 4 & 0 \\ 5 & 7 & 12 & -1 \\ 1 & 1 & 5 & 2 \\ 5 & -1 & -1 & 2 \end{bmatrix}$, compute the spectral radius of B_{J}, B_{GS}, and B_{SOR}. Will any of the iterative methods converge. Try one and see what happens.

20.21

a. Write a function `x = poisson1dj(f, n, numiter)` that uses the Jacobi iterations to numerically approximate the solution to the one-dimensional Poisson equation 20.26, where f is the right-hand side, n is the number of subintervals of $0 \leq x \leq 1$, and numiter is the number of iterations.

b. Do part (a) using the Gauss-Seidel iteration by writing a function `poisson1dgs`. If you have done Problem 20.14, use red-black ordering.

c. The exact solution to the Poisson equation

$$\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = -x(x-1), \quad u(0) = u(1) = 0$$

is

$$u(x) = \frac{x^4}{12} - \frac{x^3}{6} + \frac{x}{12}.$$

Using $n = 25$ and numiter $= 700$ (Jacobi), numiter $= 350$ (GS) compute the solution using `poisson1dj` and `poisson1dgs`. Compare the results to the true solution.

20.22 Let R be the unit square, $0 \leq x, y \leq 1$. Consider the two-dimensional Laplace equation

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \quad 0 < x,\ y < 1, \tag{20.27}$$

$$u = g\,(x,\ y) \quad \text{on } \partial R. \tag{20.28}$$

a. Write a MATLAB `function [x y u] = laplacej(g,n,maxiter)` that uses the Jacobi iteration to approximate the solution to Equation 20.27 and creates a surface plot.

b. The exact solution with

$$g\,(x,\ y) = \begin{cases} 0 & y = 0 \\ \sin(\pi x) & y = 1 \\ 0 & x = 0 \\ 0 & x = 1 \end{cases}$$

is

$$ut\,(x, y) = \frac{1}{\sinh(\pi)} \sin(\pi x) \sinh(\pi y)\,.$$

Plot the solution using the MATLAB function `ezsurf`.

c. Plot the solution by applying `laplacej` with $n = 25$ and numiter $= 100$.

20.23

a. Write function `jacobiConverge(A)` that returns true if the Jacobi iteration converges for a system with coefficient matrix A and false if it does not converge

b. Do part (a) for the Gauss-Seidel method by writing a function `gsConverge(A)`.

c. For each matrix, use your functions to determine if the Jacobi or Gauss-Seidel iterations converge.

$$A = \begin{bmatrix} 3 & 1 & 1 \\ 2 & 5 & 2 \\ 1 & 5 & 7 \end{bmatrix}, \quad B = \begin{bmatrix} 5 & -1 & 4 \\ 1 & 3 & 1 \\ 2 & 3 & 4 \end{bmatrix}, \quad C = \begin{bmatrix} 6 & 1 & 2 \\ 1 & 1 & 8 \\ 1 & 2 & 5 \end{bmatrix}, \quad D = \begin{bmatrix} 2 & 1 & 1 \\ 3 & -4 & 1 \\ -1 & 3 & 4 \end{bmatrix}$$

$$E = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 2 & 3 & 3 & 3 & 3 & 3 & 3 \\ 1 & 2 & 3 & 4 & 4 & 4 & 4 & 4 \\ 1 & 2 & 3 & 4 & 5 & 5 & 5 & 5 \\ 1 & 2 & 3 & 4 & 5 & 6 & 6 & 6 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 7 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

20.24

a. Write a function `diagDom(A)` that determines if A is strictly row diagonally dominant by returning true or false.

b. Apply your function to the matrices

i. $A = \begin{bmatrix} 3 & -1 & 1 \\ 2 & 5 & 2 \\ -8 & 3 & 12 \end{bmatrix}$

ii. $A = \begin{bmatrix} 1.0 & 0.3 & 0.4 & 0.1 & 0.1 \\ 0.00 & 2.8 & 2.00 & 0.10 & 0.40 \\ .30 & -0.10 & 0.70 & 0.20 & 0.05 \\ 0.00 & 0.60 & 1.20 & 4.25 & 2.40 \\ 0.50 & 6.80 & 1.40 & 2.30 & -10.6 \end{bmatrix}$

iii. $A = \begin{bmatrix} 3.0 & 1.0 & 1.0 & 0.90 \\ 2.0 & 8.0 & 1.0 & -4.0 \\ 5.0 & 1.0 & 7.0 & 0.80 \\ 1.0 & 1.0 & -3.0 & 5.0 \end{bmatrix}$

c. The MATLAB function call `gallery('neumann',25)` returns a sparse 25×25 matrix that results from discretizing a partial differential using a five-point finite difference operator on a regular mesh. Execute `diagDom` using the matrix

```
A = full(gallery('neumann', 25)) + diag(0.1*ones(25,1));
```

iii. $A = \begin{bmatrix} 3.0 & 1.0 & 1.0 & 0.90 \\ 2.0 & 8.0 & 1.0 & -4.0 \\ 5.0 & 1.0 & 7.0 & 0.80 \\ 1.0 & 1.0 & -3.0 & 5.0 \end{bmatrix}$

c. The MATLAB function call `gallery('neumann',25)` returns a sparse 25×25 matrix that results from discretizing a partial differential using a five-point finite difference operator on a regular mesh. Execute `diagDom` using the matrix

```
A = full(gallery('neumann', 25)) + diag(0.1*ones(25,1));
```

20.25

a. If A is a sparse matrix of the form

$$A = \begin{bmatrix} c_1 & d_1 & e_1 & & & & \\ b_1 & c_2 & d_2 & e_2 & & & \\ a_1 & b_2 & c_3 & d_3 & \ddots & & \\ & a_2 & b_3 & \ddots & \ddots & e_{n-3} & \\ & & \ddots & \ddots & \ddots & d_{n-2} & e_{n-2} \\ & & & a_{n-3} & b_{n-2} & c_{n-1} & d_{n-1} \\ & & & & a_{n-2} & b_{n-1} & c_n \end{bmatrix},$$

what is the maximum number of nonzero elements it contains? What is the density, where density $= \frac{\text{number of nonzero elements}}{n}$. Such a matrix is termed *pentadiagonal*.

b. Code a function

```
[x,iter] = pentsolve(a,b,c,d,e,x0,rhs,tol,maxiter)
```

that applies the Gauss-Seidel iteration to solve a system $Ax = rhs$. Your iteration must use only the elements on the five diagonals. Apply the termination criteria

$$\frac{\|\text{xnew} - \text{xprev}\|_2}{\|\text{xprev}\|_2} < \text{tol}.$$

c. Test your function by solving the 1000×1000 problem

$$\begin{bmatrix} 11 & -4 & -1 & 0 & 0 & \cdots & 0 \\ -4 & 11 & -4 & -1 & 0 & \cdots & 0 \\ -1 & -4 & 11 & -4 & -1 & \cdots & 0 \\ 0 & -1 & -4 & 11 & -4 & \ddots & 0 \\ 0 & 0 & -1 & -4 & 11 & \ddots & \vdots \\ & & & \ddots & \ddots & \ddots & -1 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & -4 \\ 0 & 0 & 0 & \cdots & -1 & -4 & 11 \end{bmatrix} x = rhs,$$

where *rhs* is a random matrix generated with the MATLAB command `rhs = 100*rand(1000,1)`. Time the computation. Create the matrix in part (b) using MATLAB, and solve the system using `xmat = A\rhs` and compare the two solutions and the time required.

20.26

a. Develop a MATLAB function `findomega` that plots the number of iterations of the SOR method required as a function of the relaxation parameter ω for the system $Ax = b$, $b = \text{rand}\,(n, 1)$. For the purpose of graphing, let ω range from 0.01 to 1.99 in steps of 0.005. Use $\text{tol} = 1.0 \times 10^{-12}$ and $\text{maxiter} = 1000$. Estimate the optimal value of ω by determining the ω corresponding to the minimum number of iterations required to satisfy the error tolerance. Be sure to exclude values of -1 (iteration failure) in the search for the minimum.

b. Apply findomega to the matrix E of Problem 20.23(c).

20.27

a. Develop a numerical solution to the two-dimensional Poisson problem using the red-black scheme of Problem 20.15. Name the function `poisson2dgs`.

b. Apply `poisson2dgs` to the problem of Example 20.8.

c. A thin membrane is stretched over a wire bent in the shape of a triangle. The resulting structure satisfies Laplace's equation. Consider the specific problem

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0,$$

$$u(x, y) = \begin{cases} x, & 0 \le x \le \frac{1}{2}, \quad y = 0 \\ 1 - x, & \frac{1}{2} \le x \le 1, \quad y = 0 \\ 0, & 0 \le x \le 1, \quad y = 1 \\ 0, & x = 0, \quad 0 \le y \le 1 \\ 0, & x = 1, \quad 0 \le y \le 1 \end{cases}$$

whose exact solution is

$$u(x, y) = \frac{4}{\pi^2} \sum_{n=0}^{\infty} (-1)^n \frac{\sin((2n+1)\pi x) \sinh((2n+1)\pi (1-y))}{(2n+1)^2 \sinh((2n+1)\pi)}.$$

i. Graph $u(x, y), 1 \le x, y \le 1$, by summing the series until the relative error of the partial sums is less than 10^{-8}.

ii. Apply poisson2dgs and graph the approximate solution.