GOLUB - VAN LOAN 3º ed.

unknowns must proceed on a problem-by-problem basis. General scaling strategies are unreliable. It is best to scale (if at all) on the basis of what the source problem proclaims about the significance of each a_{ij}. Measurement units and data error may have to be considered.

Example 3.5.1 (Forsythe and Moler (1967, pp. 34, 40)). If

$$\begin{bmatrix} 10 & 100{,}000 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 100{,}000 \\ 2 \end{bmatrix}$$

and the equivalent row-scaled problem

$$\begin{bmatrix} .0001 & 1 \\ 1 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

are each solved using $\beta = 10, t = 3$ arithmetic, then solutions $\hat{x} = (0.00,\ 1.00)^T$ and $\hat{x} = (1.00,\ 1.00)^T$ are respectively computed. Note that $x = (1.0001\ldots,\ .9999\ldots)^T$ is the exact solution.

3.5.3 Iterative Improvement

Suppose $Ax = b$ has been solved via the partial pivoting factorization $PA = LU$ and that we wish to improve the accuracy of the computed solution $\hat{x}$. If we execute

$$\begin{aligned} &r = b - A\hat{x} \\ &\text{Solve } Ly = Pr. \\ &\text{Solve } Uz = y. \\ &x_{new} = \hat{x} + z \end{aligned} \tag{3.5.4}$$

then in exact arithmetic $Ax_{new} = A\hat{x} + Az = (b - r) + r = b$. Unfortunately, the naive floating point execution of these formulae renders an x_{new} that is no more accurate than $\hat{x}$. This is to be expected since $\hat{r} = fl(b - A\hat{x})$ has few, if any, correct significant digits. (Recall Heuristic I.) Consequently, $\hat{z} = fl(A^{-1}r) \approx A^{-1} \cdot$ noise $\approx$ noise is a very poor correction *from the standpoint of improving the accuracy of* $\hat{x}$. However, Skeel (1980) has done an error analysis that indicates when (3.5.4) gives an improved x_{new} *from the standpoint of backwards error*. In particular, if the quantity

$$\tau = \left(\| \, |A| \, |A^{-1}| \, \|_\infty\right)\left(\max_i \ (|A||x|)_i \,/ \min_i \ (|A||x|)_i\right)$$

is not too big, then (3.5.4) produces an x_{new} such that $(A + E)x_{new} = b$ for very small E. Of course, if Gaussian elimination with partial pivoting is used then the computed $\hat{x}$ already solves a nearby system. However, this may not be the case for some of the pivot strategies that are used to preserve sparsity. In this situation, the *fixed precision iterative improvement*

step (3.5.4) can be very worthwhile and cheap. See Arioli, Demmel, and Duff (1988).

For (3.5.4) to produce a more accurate x, it is necessary to compute the residual $b - A\hat{x}$ with extended precision floating point arithmetic. Typically, this means that if t-digit arithmetic is used to compute $PA = LU$, x, y, and z, then $2t$-digit arithmetic is used to form $b - A\hat{x}$, i.e., double precision. The process can be iterated. In particular, once we have computed $PA = LU$ and initialize $x = 0$, we repeat the following:

$$\begin{aligned} &r = b - Ax \text{ (Double Precision)} \\ &\text{Solve } Ly = Pr \text{ for } y. \\ &\text{Solve } Uz = y \text{ for } z. \\ &x = x + z \end{aligned} \tag{3.5.5}$$

We refer to this process as *mixed precision iterative improvement*. The original A must be used in the double precision computation of r. The basic result concerning the performance of (3.5.5) is summarized in the following heuristic:

Heuristic III. If the machine precision $\mathbf{u}$ and condition satisfy $\mathbf{u} = 10^{-d}$ and $\kappa_\infty(A) \approx 10^q$, then after k executions of (3.5.5), x has approximately $\min(d, k(d - q))$ correct digits.

Roughly speaking, if $\mathbf{u}\kappa_\infty(A) \le 1$, then iterative improvement can ultimately produce a solution that is correct to full (single) precision. Note that the process is relatively cheap. Each improvement costs $O(n^2)$, to be compared with the original $O(n^3)$ investment in the factorization $PA = LU$. Of course, no improvement may result if A is badly enough conditioned with respect to the machine precision.

The primary drawback of mixed precision iterative improvement is that its implementation is somewhat machine-dependent. This discourages its use in software that is intended for wide distribution. The need for retaining an original copy of A is another aggravation associated with the method.

On the other hand, mixed precision iterative improvement is usually very easy to implement on a given machine that has provision for the accumulation of inner products, i.e., provision for the double precision calculation of inner products between the rows of A and x. In a short mantissa computing environment the presence of an iterative improvement routine can significantly widen the class of solvable $Ax = b$ problems.

Example 3.5.2 If (3.5.5) is applied to the system

$$\begin{bmatrix} .986 & .579 \\ .409 & .237 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} .235 \\ .107 \end{bmatrix}$$

and $\beta = 10$ and $t = 3$, then iterative improvement produces the following sequence of computed solutions:

$$\hat{x} = \begin{bmatrix} 2.11 \\ -3.17 \end{bmatrix}, \begin{bmatrix} 1.99 \\ -2.99 \end{bmatrix}, \begin{bmatrix} 2.00 \\ -3.00 \end{bmatrix}, \ldots$$

EXACT SOL.: $x = [2 \ -3]^T$

10.1.1 The Jacobi and Gauss-Seidel Iterations

Perhaps the simplest iterative scheme is the *Jacobi iteration*. It is defined for matrices that have nonzero diagonal elements. The method can be motivated by rewriting the 3-by-3 system $Ax = b$ as follows:

$$\begin{aligned} x_1 &= (b_1 - a_{12}x_2 - a_{13}x_3)/a_{11} \\ x_2 &= (b_2 - a_{21}x_1 - a_{23}x_3)/a_{22} \\ x_3 &= (b_3 - a_{31}x_1 - a_{32}x_2)/a_{33} \end{aligned}$$

Suppose $x^{(k)}$ is an approximation to $x = A^{-1}b$. A natural way to generate a new approximation $x^{(k+1)}$ is to compute

$$\begin{aligned} x_1^{(k+1)} &= (b_1 - a_{12}x_2^{(k)} - a_{13}x_3^{(k)})/a_{11} \\ x_2^{(k+1)} &= (b_2 - a_{21}x_1^{(k)} - a_{23}x_3^{(k)})/a_{22} \\ x_3^{(k+1)} &= (b_3 - a_{31}x_1^{(k)} - a_{32}x_2^{(k)})/a_{33} \end{aligned} \qquad (10.1.1)$$

This defines the Jacobi iteration for the case $n = 3$. For general n we have

```
for $i = 1{:}n$
    $x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) \Big/ a_{ii}$    (10.1.2)
end
```

Note that in the Jacobi iteration one does not use the most recently available information when computing $x_i^{(k+1)}$. For example, $x_1^{(k)}$ is used in the calculation of $x_2^{(k+1)}$ even though component $x_1^{(k+1)}$ is known. If we revise the Jacobi iteration so that we always use the most current estimate of the exact x_i then we obtain

```
for $i = 1{:}n$
    $x_i^{(k+1)} = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) \Big/ a_{ii}$    (10.1.3)
end
```

This defines what is called the *Gauss-Seidel iteration*.

For both the Jacobi and Gauss-Seidel iterations, the transition from $x^{(k)}$ to $x^{(k+1)}$ can be succinctly described in terms of the matrices L, D, and U defined by:

$$L = \begin{bmatrix} 0 & 0 & \cdots & \cdots & 0 \\ a_{21} & 0 & \cdots & & \vdots \\ a_{31} & a_{32} & \ddots & & 0 \\ \vdots & & & 0 & 0 \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & 0 \end{bmatrix}$$

$$D = \operatorname{diag}(a_{11}, \ldots, a_{nn}) \qquad (10.1.4)$$

$$U = \begin{bmatrix} 0 & a_{12} & \cdots & \cdots & a_{1n} \\ 0 & 0 & \cdots & & \vdots \\ 0 & 0 & \ddots & & a_{n-2,n} \\ \vdots & & & & a_{n-1,n} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

In particular, the Jacobi step has the form $M_J x^{(k+1)} = N_J x^{(k)} + b$ where $M_J = D$ and $N_J = -(L+U)$. On the other hand, Gauss-Seidel is defined by $M_G x^{(k+1)} = N_G x^{(k)} + b$ with $M_G = (D+L)$ and $N_G = -U$.

10.1.2 Splittings and Convergence

The Jacobi and Gauss-Seidel procedures are typical members of a large family of iterations that have the form

$$Mx^{(k+1)} = Nx^{(k)} + b \qquad (10.1.5)$$

where $A = M - N$ is a *splitting* of the matrix A. For the iteration (10.1.5) to be practical, it must be "easy" to solve a linear system with M as the matrix. Note that for Jacobi and Gauss-Seidel, M is diagonal and lower triangular respectively.

Whether or not (10.1.5) converges to $x = A^{-1}b$ depends upon the eigenvalues of $M^{-1}N$. In particular, if the *spectral radius* of an n-by-n matrix G is defined by

$$\rho(G) = \max\{\, |\lambda| : \lambda \in \lambda(G) \,\},$$

then it is the size of $\rho(M^{-1}N)$ is critical to the success of (10.1.5).

Theorem 10.1.1 *Suppose $b \in \mathbb{R}^n$ and $A = M - N \in \mathbb{R}^{n\times n}$ is nonsingular. If M is nonsingular and the spectral radius of $M^{-1}N$ satisfies the inequality $\rho(M^{-1}N) < 1$, then the iterates $x^{(k)}$ defined by $Mx^{(k+1)} = Nx^{(k)} + b$ converge to $x = A^{-1}b$ for any starting vector $x^{(0)}$.*

Proof. Let $e^{(k)} = x^{(k)} - x$ denote the error in the kth iterate. Since $Mx = Nx + b$ it follows that $M(x^{(k+1)} - x) = N(x^{(k)} - x)$, and thus, the error in $x^{(k+1)}$ is given by $e^{(k+1)} = M^{-1}Ne^{(k)} = (M^{-1}N)^{k+1}e^{(0)}$. From Lemma 7.3.2 we know that $(M^{-1}N)^k \to 0$ iff $\rho(M^{-1}N) < 1$. □

This result is central to the study of iterative methods where algorithmic development typically proceeds along the following lines:

- A splitting $A = M - N$ is proposed where linear systems of the form $Mz = d$ are "easy" to solve.

- Classes of matrices are identified for which the iteration matrix $G = M^{-1}N$ satisfies $\rho(G) < 1$.
- Further results about $\rho(G)$ are established to gain intuition about how the error $e^{(k)}$ tends to zero.

For example, consider the Jacobi iteration, $Dx^{(k+1)} = -(L+U)x^{(k)} + b$. One condition that guarantees $\rho(M_J^{-1}N_J) < 1$ is strict diagonal dominance. Indeed, if A has that property (defined in §3.4.10), then

$$\rho(M_J^{-1}N_J) \le \| D^{-1}(L+U) \|_\infty = \max_{1\le i\le n} \sum_{\substack{j=1 \\ j\ne i}}^{n} \left|\frac{a_{ij}}{a_{ii}}\right| < 1$$

Usually, the "more dominant" the diagonal the more rapid the convergence but there are counterexamples. See P10.1.7.

A more complicated spectral radius argument is needed to show that Gauss-Seidel converges for symmetric positive definite A.

Theorem 10.1.2 *If $A \in \mathbb{R}^{n\times n}$ is symmetric and positive definite, then the Gauss-Seidel iteration (10.1.3) converges for any $x^{(0)}$.*

Proof. Write $A = L + D + L^T$ where $D = \text{diag}(a_{ii})$ and L is strictly lower triangular. In light of Theorem 10.1.1 our task is to show that the matrix $G = -(D+L)^{-1}L^T$ has eigenvalues that are inside the unit circle. Since D is positive definite we have $G_1 = D^{1/2}GD^{-1/2} = -(I+L_1)^{-1}L_1^T$, where $L_1 = D^{-1/2}LD^{-1/2}$. Since G and G_1 have the same eigenvalues, we must verify that $\rho(G_1) < 1$. If $G_1x = \lambda x$ with $x^Hx = 1$, then we have $-L_1^Tx = \lambda(I+L_1)x$ and thus, $-x^HL_1^Tx = \lambda(1 + x^HL_1x)$. Letting $a + bi = x^HL_1x$ we have

$$|\lambda|^2 = \left|\frac{-a+bi}{1+a+bi}\right|^2 = \frac{a^2+b^2}{1+2a+a^2+b^2}.$$

However, since $D^{-1/2}AD^{-1/2} = I + L_1 + L_1^T$ is positive definite, it is not hard to show that $0 < 1 + x^HL_1x + x^HL_1^Tx = 1 + 2a$ implying $|\lambda| < 1$. □

This result is frequently applicable because many of the matrices that arise from discretized elliptic PDE's are symmetric positive definite. Numerous other results of this flavor appear in the literature.

10.1.3 Practical Implementation of Gauss-Seidel

We now focus on some practical details associated with the Gauss-Seidel iteration. With overwriting the Gauss-Seidel step (10.1.3) is particularly simple to implement:

```
for $i = 1:n$
    $x_i = \left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j - \sum_{j=i+1}^{n} a_{ij}x_j\right) \Big/ a_{ii}$
end
```

This computation requires about twice as many flops as there are nonzero entries in the matrix A. It makes no sense to be more precise about the work involved because the actual implementation depends greatly upon the structure of the problem at hand.

In order to stress this point we consider the application of (10.1.3) to the NM-by-NM block tridiagonal system

$$\begin{bmatrix} T & -I_N & & \cdots & 0 \\ -I_N & T & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & -I_N \\ 0 & \cdots & & -I_N & T \end{bmatrix} \begin{bmatrix} g_1 \\ g_2 \\ \vdots \\ \vdots \\ g_M \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ \vdots \\ f_M \end{bmatrix} \qquad (10.1.6)$$

where

$$T = \begin{bmatrix} 4 & -1 & & \cdots & 0 \\ -1 & 4 & \ddots & & \vdots \\ & \ddots & \ddots & \ddots & \\ \vdots & & \ddots & \ddots & -1 \\ 0 & \cdots & & -1 & 4 \end{bmatrix}, \quad g_j = \begin{bmatrix} G(1,j) \\ G(2,j) \\ \vdots \\ \vdots \\ G(N,j) \end{bmatrix}, \quad f_j = \begin{bmatrix} F(1,j) \\ F(2,j) \\ \vdots \\ \vdots \\ F(N,j) \end{bmatrix}.$$

This problem arises when the Poisson equation is discretized on a rectangle. It is easy to show that the matrix A is positive definite.

With the convention that $G(i,j) = 0$ whenever $i \in \{0, N+1\}$ or $j \in \{0, M+1\}$ we see that with overwriting the Gauss-Seidel step takes on the form:

```
for $j = 1:M$
    for $i = 1:N$
        $G(i,j) = (F(i,j) + G(i-1,j) + G(i+1,j) +$
                  $G(i,j-1) + G(i,j+1))/4$
    end
end
```

Note that in this problem no storage is required for the matrix A.

10.1.4 Successive Over-Relaxation

The Gauss-Seidel iteration is very attractive because of its simplicity. Unfortunately, if the spectral radius of $M_G^{-1}N_G$ is close to unity, then it may be prohibitively slow because the error tends to zero like $\rho(M_G^{-1}N_G)^k$. To rectify this, let $\omega \in \mathbb{R}$ and consider the following modification of the Gauss-Seidel step:

```
for $i = 1{:}n$
    $x_i^{(k+1)} = \omega\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right) \Big/ a_{ii}$
        $+ (1-\omega)x_i^{(k)}$        (10.1.7)
end
```

This defines the method of *successive over-relaxation* (SOR). Using (10.1.4) we see that in matrix terms, the SOR step is given by

$$M_\omega x^{(k+1)} = N_\omega x^{(k)} + \omega b \qquad (10.1.8)$$

where $M_\omega = D + \omega L$ and $N_\omega = (1-\omega)D - \omega U$. For a few structured (but important) problems such as (10.1.6), the value of the *relaxation parameter* ω that minimizes $\rho(M_\omega^{-1}N_\omega)$ is known. Moreover, a significant reduction in $\rho(M_1^{-1}N_1) = \rho(M_G^{-1}N_G)$ can result. In more complicated problems, however, it may be necessary to perform a fairly sophisticated eigenvalue analysis in order to determine an appropriate ω.

10.1.5 The Chebyshev Semi-Iterative Method

Another way to accelerate the convergence of an iterative method makes use of Chebyshev polynomials. Suppose $x^{(1)}, \ldots, x^{(k)}$ have been generated via the iteration $Mx^{(j+1)} = Nx^{(j)} + b$ and that we wish to determine coefficients $\nu_j(k)$, $j = 0{:}k$ such that

$$y^{(k)} = \sum_{j=0}^{k} \nu_j(k) x^{(j)} \qquad (10.1.9)$$

represents an improvement over $x^{(k)}$. If $x^{(0)} = \cdots = x^{(k)} = x$, then it is reasonable to insist that $y^{(k)} = x$. Hence, we require

$$\sum_{j=0}^{k} \nu_j(k) = 1. \qquad (10.1.10)$$

Subject to this constraint, we consider how to choose the $\nu_j(k)$ so that the error in $y^{(k)}$ is minimized.

Recalling from the proof of Theorem 10.1.1 that $x^{(k)} - x = (M^{-1}N)^k e^{(0)}$ where $e^{(0)} = x^{(0)} - x$, we see that

$$y^{(k)} - x = \sum_{j=0}^{k} \nu_j(k)(x^{(j)} - x) = \sum_{j=0}^{k} \nu_j(k)(M^{-1}N)^j e^{(0)}.$$

Working in the 2-norm we therefore obtain

$$\| y^{(k)} - x \|_2 \le \| p_k(G) \|_2 \| e^{(0)} \|_2 \qquad (10.1.11)$$

where $G = M^{-1}N$ and

$$p_k(z) = \sum_{j=0}^{k} \nu_j(k) z^j.$$

Note that the condition (10.1.10) implies $p_k(1) = 1$.

At this point we assume that G is symmetric with eigenvalues λ_i that satisfy $-1 < \alpha \le \lambda_n \le \cdots \le \lambda_1 \le \beta < 1$. It follows that

$$\| p_k(G) \|_2 = \max_{\lambda_i \in \lambda(A)} |p_k(\lambda_i)| \le \max_{\alpha \le \lambda \le \beta} |p_k(\lambda)|.$$

Thus, to make the norm of $p_k(G)$ small, we need a polynomial $p_k(z)$ that is small on $[\alpha, \beta]$ subject to the constraint that $p_k(1) = 1$.

Consider the Chebyshev polynomials $c_j(z)$ generated by the recursion $c_j(z) = 2zc_{j-1}(z) - c_{j-2}(z)$ where $c_0(z) = 1$ and $c_1(z) = z$. These polynomials satisfy $|c_j(z)| \le 1$ on [-1, 1] but grow rapidly off this interval. As a consequence, the polynomial

$$p_k(z) = \frac{c_k\left(-1 + 2\dfrac{z-\alpha}{\beta-\alpha}\right)}{c_k(\mu)}$$

where

$$\mu = -1 + 2\frac{1-\alpha}{\beta-\alpha} = 1 - 2\frac{1-\beta}{\beta-\alpha}$$

satisfies $p_k(1) = 1$ and tends to be small on $[\alpha, \beta]$. From the definition of $p_k(z)$ and equation (10.1.11) we see

$$\| y^{(k)} - x \|_2 \le \frac{\| x - x^{(0)} \|_2}{|c_k(\mu)|}.$$

Thus, the larger μ is, the greater the acceleration of convergence.

In order for the above to be a practical acceleration procedure, we need a more efficient method for calculating $y^{(k)}$ than (10.1.9). We have been