CHECKING CONVERGENCE

Checking convergence of an iteration scheme is not always an easy task. In this situation, we have the recursive error formula

$$x - x^{(m+1)} = M\left[x - x^{(m)}\right], \qquad m \geq 0 \qquad (1)$$

Replace m by $m-1$, obtaining

$$x - x^{(m)} = M\left[x - x^{(m-1)}\right], \qquad m \geq 1$$

Subtract the first from the second, obtaining

$$x^{(m+1)} - x^{(m)} = M\left[x^{(m)} - x^{(m-1)}\right], \qquad m \geq 1$$

If $\|M\| < 1$ for some operator matrix norm, we can write

$$\left\|x^{(m+1)} - x^{(m)}\right\| \leq \|M\| \left\|x^{(m)} - x^{(m-1)}\right\|$$

$$\frac{\left\|x^{(m+1)} - x^{(m)}\right\|}{\left\|x^{(m)} - x^{(m-1)}\right\|} \leq \|M\|$$

$$\frac{\left\| x^{(m+1)} - x^{(m)} \right\|}{\left\| x^{(m)} - x^{(m-1)} \right\|} \leq \|M\|$$

We will often attempt to estimate $c \equiv \|M\|$ by using these ratios of the norms of the successive differences. Often, but not always, these ratios are approximately constant as m increases. Thus we use

$$c \approx c_m = \frac{\left\| x^{(m+1)} - x^{(m)} \right\|}{\left\| x^{(m)} - x^{(m-1)} \right\|}$$

or

$$c \approx c_m = \sup_{m-p \leq q \leq m} \frac{\left\| x^{(q+1)} - x^{(q)} \right\|}{\left\| x^{(q)} - x^{(q-1)} \right\|}$$

for some $p > 0$, or perhaps the geometric average of several such successive ratios.

Thus we assume we know $c < 1$ for which (1) is valid. How do we then estimate the error?

$$\begin{aligned}
\left\|x^{(m+1)} - x^{(m)}\right\| &= \left\|\left[x - x^{(m)}\right] - \left[x - x^{(m+1)}\right]\right\| \\
&\geq \left\|\left[x - x^{(m)}\right]\right\| - \left\|\left[x - x^{(m+1)}\right]\right\| \\
&\geq \frac{1}{c}\left\|\left[x - x^{(m+1)}\right]\right\| - \left\|\left[x - x^{(m+1)}\right]\right\| \\
&= \frac{1-c}{c}\left\|\left[x - x^{(m+1)}\right]\right\| \\
\left\|\left[x - x^{(m+1)}\right]\right\| &\leq \frac{c}{1-c}\left\|x^{(m+1)} - x^{(m)}\right\|
\end{aligned}$$

In general, this is a very reasonable error bound. We will see it illustrated in a later section with other iteration methods.

EXAMPLE

The above ideas are illustrated in Table 8.5 of the text, on page 553, for Gauss-Seidel iteration. The system being solved is from discretizing the problem

$$\begin{aligned} \Delta u(x,y) &= 2\left(x^2+y^2\right), && (x,y) \in R \\ u(x,y) &= x^2y^2, && (x,y) \in \Gamma \end{aligned} \tag{2}$$

with $N = 16$ subdivisions of $[0,1]$ for each of the variables x and y. The true solution is $u(x,y) = x^2y^2$. In this case, the discretization error is exactly zero. Therefore, the solution of the linear system is known exactly, namely

$$u_h(x_j, y_k) = x_j^2 y_k^2, \qquad 0 \le j,k \le N$$

Thus we can measure the iteration error exactly, to check on the accuracy of our predicted error bounds. From the table, the estimated error bound is quite close to the actual error $\left\| \left[x - x^{(m+1)} \right] \right\|_\infty$. Note also the slow rate of convergence, with the values of *Ratio* approaching 0.962.

COST COMPARISONS

How does the cost of iteration compare to that of Gaussian elimination, assuming the latter is even possible for the system $Ax = b$ being solved. We recall that Gaussian elimination requires around $\frac{2}{3}n^3$ arithmetic operations for a system of order n.

With Gauss-Jacobi and Gauss-Seidel iteration, there are around $2n^2$ arithmetic operations per iteration, assuming that A is dense. For the system obtained by discretizing the Poisson equation, the cost of a single iteration is approximately $5n$ arithmetic operations. Let $\nu(n)$ denote the operations count per iteration. To compare further, you need to know how many iterates are to be computed. To do this, we need to make an assumption on the rate of convergence.

Assume that the iterates $\left\{x^{(m)}\right\}$ satisfy

$$\left\|x - x^{(m+1)}\right\| \leq c \left\|x - x^{(m)}\right\|, \qquad m \geq 0 \qquad (3)$$

for some $c < 1$. Further assume that we want to find an iterate which will satisfy

$$\left\|x - x^{(m)}\right\| \leq \varepsilon \left\|x - x^{(0)}\right\| \qquad (4)$$

If $x^{(0)} = 0$, as often happens, then this test becomes the relative error test

$$\frac{\left\|x - x^{(m)}\right\|}{\|x\|} \leq \varepsilon$$

Using (3),

$$\left\|x - x^{(m)}\right\| \leq c^m \left\|x - x^{(0)}\right\|$$

Then (4) is satisfied if we require

$$c^m \leq \varepsilon$$

$$m \geq \frac{-\log \varepsilon}{R(c)} \equiv m^*, \qquad R(c) = -\log c$$

Thus the total cost of solving $Ax = b$ is

$$\text{Cost}(c, \varepsilon, n) = m^* \nu(n)$$

Question: When is

$$\text{Cost}(c, \varepsilon, n) \leq \frac{2}{3} n^3 \, ?$$

Assuming $\nu(n) = 2n^2$, as when A is dense, we have iteration is more efficient than Gaussian elimination if

$$\begin{aligned} m^* \left(2n^2\right) &\leq \tfrac{2}{3} n^3 \\ m^* \equiv \tfrac{-\log \varepsilon}{R(c)} &\leq \tfrac{1}{3} n \end{aligned}$$

For solving to machine precision, we would take $\varepsilon \approx \mathbf{u}$, the unit round. Then Gaussian elimination is more efficient unless c is sufficiently close to zero, meaning the iteration converges sufficiently rapidly. For the cross-over point as to which is the more efficient procedure, use the equation

$$\frac{n}{3} = \frac{-\log \mathbf{u}}{R(c)}$$

to determine n from c. Denote the break-even value of n by n_c. If $n \geq n_c$, then iteration is more efficient than Gaussian elimination.

In the following table, we use $\mathbf{u} = 2.22 \times 10^{-16}$, the unit round in double precision for IEEE arithmetic.

c	m^*	n_c
.1	15.7	48
.2	22.4	68
.4	39.3	118
.6	70.6	212
.8	161.5	485
.9	342.1	1027
.99	3586	10758

If $n \geq n_c$, then iteration is more efficient than Gaussian elimination. Again, recall that this is for the case in which A is dense, so that $\nu(n) = 2n^2$.

Slow convergence. Let $c = 1 - \delta$ for some positive $\delta \approx 0$. Then

$$R(c) = -\log(1 - \delta) \approx \delta$$

Then the cost becomes

$$\text{Cost}(c, \varepsilon, n) = m^* \nu(n) \approx \frac{-\log \varepsilon}{\delta} \nu(n)$$

Then doubling δ to 2δ will halve the total cost.

This is of importance for linear systems $Ax = b$ that arise from discretization of Poisson's equation. In that case, $\delta \to 0$ as $N \to \infty$. We return to this in §8.8.

ACCELERATION OF CONVERGENCE

As with other subjects in numerical analysis, one of the reasons for analyzing and understanding the error behaviour of an algorithm is to use that information to accelerate the convergence. We have not yet done that with the Gauss-Seidel method, but will state an acceleration method for it. Historically, the method preceded by a number of years an understanding of the acceleration.

$$\begin{aligned} z_i^{(m+1)} &= \frac{1}{a_{ii}}\left[b_i - \sum_{j=1}^{i-1} a_{i,j}x_j^{(m+1)} - \sum_{j=i+1}^{n} a_{i,j}x_j^{(m)}\right] \\ x_i^{(m+1)} &= \omega z_i^{(m+1)} + (1-\omega)x_i^{(m)} \end{aligned} \tag{5}$$

for $i = 1, ..., n$ and $m = 0, 1, ...$ The number ω is called the acceleration parameter. If $\omega = 1$, we have the ordinary Gauss-Seidel iteration.

One could accelerate the vectors $\left\{x^{(m)}\right\}$, but in this case each component is being accelerated as soon as it is computed. With an optimal choice of ω, this iteration will converge much more rapidly than will the Gauss-Seidel method. Finding that optimal value can be difficult, however. In the next section, we look at the particular case of the discretization of the Poisson equation.

As an example, see Table 8.7 in the text, on page 556. It is the acceleration of the earlier example (2) for the Poisson equation. Note the less regular behaviour of the values of *Ratio*, making it more difficult to bound the error with confidence.