For the **SOR method**, we have $\boldsymbol{Q} = 1/\omega(\boldsymbol{D} - \omega\boldsymbol{C}_L)$, so

$$\mathcal{L}_\omega = \boldsymbol{I} - \boldsymbol{Q}^{-1}\boldsymbol{A} = (\boldsymbol{D} - \omega\boldsymbol{C}_L)^{-1}[\omega\boldsymbol{C}_U + (1-\omega)\boldsymbol{D}]$$
$$\boldsymbol{h} = \boldsymbol{Q}^{-1}\boldsymbol{b} = \omega(\boldsymbol{D} - \omega\boldsymbol{C}_L)^{-1}\boldsymbol{b}$$

Another View of Overrelaxation

In some cases, the rate of convergence of the basic iterative scheme (2) can be improved by the introduction of an auxiliary vector and an *acceleration parameter* ω as follows:

$$\boldsymbol{Q}\boldsymbol{z}^{(k)} = (\boldsymbol{Q} - \boldsymbol{A})x^{(k-1)} + \boldsymbol{b}$$
$$\boldsymbol{x}^{(k)} = \omega\boldsymbol{z}^{(k)} + (1-\omega)\boldsymbol{x}^{(k-1)}$$

or

$$\boldsymbol{x}^{(k)} = \omega\left\{(\boldsymbol{I} - \boldsymbol{Q}^{-1}\boldsymbol{A})\boldsymbol{x}^{(k-1)} + \boldsymbol{Q}^{-1}\boldsymbol{b}\right\} + (1-\omega)\boldsymbol{x}^{(k-1)}$$

The parameter ω gives a weighting in favor of the updated values. When $\omega = 1$, this procedure reduces to the basic iterative method, and when $1 < \omega < 2$, the rate of convergence may be improved, which is called **overrelaxation**. When $\boldsymbol{Q} = \boldsymbol{D}$, we have the **Jacobi overrelaxation (JOR) method**:

$$\boldsymbol{x}^{(k)} = \omega\left\{\boldsymbol{B}\boldsymbol{x}^{(k-1)} + \boldsymbol{h}\right\} + (1-\omega)\boldsymbol{x}^{(k-1)}$$

Overrelaxation has particular advantages when used with the Gauss-Seidel method in a slightly different way:

$$\boldsymbol{D}\boldsymbol{z}^{(k)} = \boldsymbol{C}_L\boldsymbol{x}^{(k)} + \boldsymbol{C}_U\boldsymbol{x}^{(k-1)} + \boldsymbol{b}$$
$$\boldsymbol{x}^{(k)} = \omega\boldsymbol{z}^{(k)} + (1-\omega)\boldsymbol{x}^{(k-1)}$$

and we have the **SOR method**:

$$\boldsymbol{x}^{(k)} = \mathcal{L}_\omega\boldsymbol{x}^{(k-1)} + \boldsymbol{h}$$

Conjugate Gradient Method

The conjugate gradient method is one of the most popular iterative methods for solving sparse systems of linear equations. This is particularly true for systems that arise in the numerical solutions of partial differential equations.

We begin with a brief presentation of definitions and associated notation. (Some of them are presented more fully in Chapter 16.) Assume that the real $n \times n$ matrix $\boldsymbol{A}$ is **symmetric**, meaning that $\boldsymbol{A}^T = \boldsymbol{A}$. The **inner product** of two vectors $\boldsymbol{u} = (u_1, u_2, \ldots, u_n)$ and $\boldsymbol{v} = (v_1, v_2, \ldots, v_n)$ can be written as $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \boldsymbol{u}^T\boldsymbol{v} = \sum_{i=1}^{n} u_i v_i$, which is the scalar sum. Note that $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = \langle \boldsymbol{v}, \boldsymbol{u}\rangle$. If $\boldsymbol{u}$ and $\boldsymbol{v}$ are mutually orthogonal, then $\langle \boldsymbol{u}, \boldsymbol{v}\rangle = 0$. An **$\boldsymbol{A}$-inner product** of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ is defined as

$$\langle \boldsymbol{u}, \boldsymbol{v}\rangle_A = \langle \boldsymbol{A}\boldsymbol{u}, \boldsymbol{v}\rangle = \boldsymbol{u}^T\boldsymbol{A}^T\boldsymbol{v}$$

Two nonzero vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ are **$\boldsymbol{A}$-conjugate** if $\langle \boldsymbol{u}, \boldsymbol{v}\rangle_A = 0$. An $n \times n$ matrix $\boldsymbol{A}$ is **positive definite** if

$$\langle \boldsymbol{x}, \boldsymbol{x}\rangle_A > 0$$

for all nonzero vectors $\boldsymbol{x} \in \mathbb{R}^n$. In general, expressions such as $\langle \boldsymbol{u}, \boldsymbol{v} \rangle$ and $\langle \boldsymbol{u}, \boldsymbol{v} \rangle_A$ reduce to 1×1 matrices and are treated as scalar values. A **quadratic form** is a scalar quadratic function of a vector of the form

$$f(\boldsymbol{x}) = \frac{1}{2}\langle \boldsymbol{x}, \boldsymbol{x} \rangle_A - \langle \boldsymbol{b}, \boldsymbol{x} \rangle + c$$

Here, $\boldsymbol{A}$ is a matrix, $\boldsymbol{x}$ and $\boldsymbol{b}$ are vectors, and c is a scalar constant. The **gradient** of a quadratic form

$$f'(\boldsymbol{x}) = \left[\partial f(\boldsymbol{x})/\partial x_1, \quad \partial f(\boldsymbol{x})/\partial x_2, \quad \cdots, \quad \partial f(\boldsymbol{x})/\partial x_n \right]^T$$

We can derive the following:

$$f'(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{A}^T\boldsymbol{x} + \frac{1}{2}\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$$

If $\boldsymbol{A}$ is symmetric, this reduces to

$$f'(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}$$

Setting the gradient to zero, we obtain the linear system to be solved, $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. Therefore, the solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is a critical point of $f(\boldsymbol{x})$. If $\boldsymbol{A}$ is symmetric and positive definite, then $f(\boldsymbol{x})$ is minimized by the solution of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$. So an alternative way of solving the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ is by finding an $\boldsymbol{x}$ that minimizes $f(\boldsymbol{x})$.

We want to solve the linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

where the $n \times n$ matrix $\boldsymbol{A}$ is symmetric and positive definite.

Suppose that $\{\boldsymbol{p}^{(1)}, \boldsymbol{p}^{(2)}, \ldots, \boldsymbol{p}^{(k)}, \ldots, \boldsymbol{p}^{(n)}\}$ is a set containing a sequence of n mutually conjugate **direction vectors**. Then they form a basis for the space $\mathbb{R}^n$. Hence, we can expand the true solution vector $\boldsymbol{x}^*$ of $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ into a linear combination of these basis vectors:

$$\boldsymbol{x}^* = \alpha_1 \boldsymbol{p}^{(1)} + \alpha_2 \boldsymbol{p}^{(2)} + \cdots + \alpha^{(k)} \boldsymbol{p}^{(k)} + \cdots + \alpha_n \boldsymbol{p}^{(n)}$$

where the coefficients are given by

$$\alpha_k = \langle \boldsymbol{p}^{(k)}, \boldsymbol{b} \rangle / \langle \boldsymbol{p}^{(k)}, \boldsymbol{p}^{(k)} \rangle_A$$

This can be viewed as a direct method for solving the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$: First find the sequence of n conjugate direction vectors $\boldsymbol{p}^{(k)}$, and then compute the coefficients α_k. However, in practice, this approach is impractical because it would take too much computer time and storage.

On the other hand, if we view the conjugate gradient method as an iterative method, then we could solve large sparse linear systems in a reasonable amount of time and storage. The key is carefully choosing a small set of the conjugate direction vectors $\boldsymbol{p}^{(k)}$ so that we do not need them all to obtain a good approximation to the true solution vector.

Start with an initial guess $\boldsymbol{x}^{(0)}$ to the true solution $\boldsymbol{x}^*$. We can assume without loss of generality that $\boldsymbol{x}^{(0)}$ is the zero vector. The true solution $\boldsymbol{x}^*$ is also the unique minimizer of

$$f(\boldsymbol{x}) = \frac{1}{2}\langle \boldsymbol{x}, \boldsymbol{x} \rangle_A - \langle \boldsymbol{x}, \boldsymbol{x} \rangle = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{x}$$

for $\boldsymbol{x} \in \mathbb{R}^n$. This suggests taking the first basis vector $\boldsymbol{p}^{(1)}$ to be the gradient of f at $\boldsymbol{x} = \boldsymbol{x}^{(0)}$, which equals $-\boldsymbol{b}$. The other vectors in the basis are now conjugate to the gradient—hence

the name *conjugate gradient method*. The kth residual vector is

$$\boldsymbol{r}^{(k)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(k)}$$

The gradient descent method moves in the direction $\boldsymbol{r}^{(k)}$. Take the direction closest to the gradient vector $\boldsymbol{r}^{(k)}$ by insisting that the direction vectors $\boldsymbol{p}^{(k)}$ be conjugate to each other. Putting all this together, we obtain the expression

$$\boldsymbol{p}^{(k+1)} = \boldsymbol{r}^{(k)} - \left[\left\langle \boldsymbol{p}^{(k)}, \boldsymbol{r}^{(k)} \right\rangle_{\boldsymbol{A}} \Big/ \left\langle \boldsymbol{p}^{(k)}, \boldsymbol{p}^{(k)} \right\rangle_{\boldsymbol{A}}\right] \boldsymbol{p}_k$$

After some simplifications, the algorithm is obtained for solving the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, where the coefficient matrix $\boldsymbol{A}$ is real, symmetric, and positive definite. The input vector $\boldsymbol{x}^{(0)}$ is an initial approximation to the solution or the zero vector.

In theory, the conjugate gradient iterative method solves a system of n linear equations in at most n steps, if the matrix $\boldsymbol{A}$ is symmetric and positive definite. Moreover, the nth iterative vector $\boldsymbol{x}^{(n)}$ is the unique minimizer of the quadratic function $q(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b}$. When the conjugate gradient method was introduced by Hestenes and Stiefel [1952], the initial interest in it waned once it was discovered that this finite-termination property was not obtained in practice. But two decades later, there was renewed interest in this method when it was viewed as an iterative process by Reid [1971] and others. In practice, the solution of a system of linear equations can often be found with satisfactory precision in a number of steps considerably less than the order of the system.

Here is a pseudocode for the **conjugate gradient algorithm**:

$k \leftarrow 0$; $\boldsymbol{x} \leftarrow \boldsymbol{0}$; $\boldsymbol{r} \leftarrow \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}$; $\delta \leftarrow \langle \boldsymbol{r}, \boldsymbol{r} \rangle$
while $\left(\sqrt{\delta} > \varepsilon\sqrt{\langle \boldsymbol{b}, \boldsymbol{b} \rangle}\right)$ **and** $\left(k < k_{\max}\right)$
  $k \leftarrow k + 1$
  if $k = 1$ **then**
    $\boldsymbol{p} \leftarrow \boldsymbol{r}$
  else
    $\beta \leftarrow \delta/\delta_{\text{old}}$
    $\boldsymbol{p} \leftarrow \boldsymbol{r} + \beta\boldsymbol{p}$
  end if
  $\boldsymbol{w} \leftarrow \boldsymbol{A}\boldsymbol{p}$
  $\alpha \leftarrow \delta/\langle \boldsymbol{p}, \boldsymbol{w} \rangle$
  $\boldsymbol{x} \leftarrow \boldsymbol{x} + \alpha\boldsymbol{p}$
  $\boldsymbol{r} \leftarrow \boldsymbol{r} - \alpha\boldsymbol{w}$
  $\delta_{\text{old}} \leftarrow \delta$
  $\delta \leftarrow \langle \boldsymbol{r}, \boldsymbol{r} \rangle$
end while

Here, ε is a parameter used in the convergence criterion (such as $\varepsilon = 10^{-5}$), and $k_{\max}$ is the maximum number of iterations allowed. Usually, the number of iterations needed is much less than the size of the linear system. We save the previous value of δ in the variable δ_{old}. If a good guess for the solution vector $\boldsymbol{x}$ is known, then it should be used as an initial vector instead of zero. The variable ε is the desired convergence tolerance. The algorithm produces not only a sequence of vectors $\boldsymbol{x}^{(i)}$ that converges to the solution but an orthogonal sequence of residual vectors $\boldsymbol{r}^{(i)} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(i)}$ and an $\boldsymbol{A}$-orthogonal sequence of

search direction vectors $\boldsymbol{p}^{(i)}$, namely, $\langle \boldsymbol{r}^{(i)}, \boldsymbol{r}^{(j)} \rangle = 0$ if $i \neq j$ and $\langle \boldsymbol{p}^{(i)}, \boldsymbol{A}\boldsymbol{p}^{(j)} \rangle = 0$ if $i \neq j$. (The main computational features of the conjugate gradient algorithm are complicated to derive, but the final conclusion is that in each step, only *one* matrix-vector multiplication is required and only a few dot-products are computed. These are extremely desirable attributes in solving large and sparse linear systems. Also, unlike Gaussian elimination, there is no fill-in, so only the nonzero entries in $\boldsymbol{A}$ need to be stored in the computer memory. For some partial differential equation problems, the equations in the linear system can be represented by stencils that describe the nonzero structure within the coefficient matrix. Sometimes these stencils are used in a computer program rather than storing the nonzero entries in the coefficient matrix.

EXAMPLE 5 Use the conjugate gradient method to solve this linear system:

$$\begin{bmatrix} 2 & -1 & 0 \\ -1 & 3 & -1 \\ 0 & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 8 \\ -5 \end{bmatrix}$$

Solution Programming the pseudocode, we obtain the iterates

$$\begin{aligned} \boldsymbol{x}^{(0)} &= [0.00000,\ 0.00000,\ 0.00000]^T \\ \boldsymbol{x}^{(1)} &= [0.29221,\ 2.33766,\ -1.46108]^T \\ \boldsymbol{x}^{(2)} &= [1.82254,\ 2.60772,\ -1.55106]^T \\ \boldsymbol{x}^{(3)} &= [2.00000,\ 3.00000,\ -1.00000]^T \end{aligned}$$

In only three iterations, we have the answer accurate to full machine precision, which illustrates the finite termination property. The matrix $\boldsymbol{A}$ is symmetric positive definite and the eigenvalues of $\boldsymbol{A}$ are 1, 2, 4. This simple example may be a bit misleading because one cannot expect such rapid convergence in realistic applications. (The rate of convergence depends on various properties of the linear system.) In fact, the above example is too small to illustrate the power of advanced iterative methods on very large and sparse systems. ■

The conjugate gradient method may converge slowly when the matrix $\boldsymbol{A}$ is ill-conditioned; however, the convergence can be accelerated by a technique called **preconditioning**. This involves a matrix $\boldsymbol{M}^{-1}$ that approximates $\boldsymbol{A}$ so that $\boldsymbol{M}^{-1}\boldsymbol{A}$ is well-conditioned and $\boldsymbol{M}\boldsymbol{x} = \boldsymbol{y}$ is easily solved. For many very large and sparse linear systems, preconditioned conjugate gradient methods have now become the iterative methods of choice! For additional details, see Golub and Van Loan [1996] as well as many other standard textbooks and references.

Summary

(1) For the linear system

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$$

the general form of an iterative method is

$$\boldsymbol{x}^{(k)} = \mathcal{G}\boldsymbol{x}^{(k-1)} + \boldsymbol{h}$$