CHAPTER

8

METHODS RELATED TO THE NORMAL EQUATIONS

There are a number of techniques for converting a non-symmetric linear system into a symmetric one. One such technique solves the equivalent linear system $A^T Ax = A^T b$, called the normal equations. Often, this approach is avoided in practice because the coefficient matrix $A^T A$ is much worse conditioned than A. However, the normal equations approach may be adequate in some situations. Indeed, there are even applications in which it is preferred to the usual Krylov subspace techniques. This chapter covers iterative methods which are either directly or implicitly related to the normal equations.

8.1 THE NORMAL EQUATIONS

In order to solve the linear system $Ax = b$ when A is nonsymmetric, we can solve the equivalent system

$$A^T A\, x = A^T b \tag{8.1}$$

which is Symmetric Positive Definite. This system is known as the system of the *normal equations* associated with the least-squares problem,

$$\text{minimize} \quad \|b - Ax\|_2. \tag{8.2}$$

Note that (8.1) is typically used to solve the least-squares problem (8.2) for *over-determined* systems, i.e., when A is a rectangular matrix of size $n \times m$, $m < n$.

A similar well known alternative sets $x = A^T u$ and solves the following equation for u:

$$AA^T u = b. \tag{8.3}$$

Once the solution u is computed, the original unknown x could be obtained by multiplying u by A^T. However, most of the algorithms we will see do not invoke the u variable explicitly and work with the original variable x instead. The above system of equations can be used to solve *under-determined* systems, i.e., those systems involving rectangular matrices of size $n \times m$, with $n < m$. It is related to (8.1) in the following way. Assume that $n \leq m$ and that A has full rank. Let x_* be *any* solution to the underdetermined system $Ax = b$. Then (8.3) represents the normal equations for the least-squares problem,

$$\text{minimize} \quad \|x_* - A^T u\|_2. \tag{8.4}$$

Since by definition $A^T u = x$, then (8.4) will find the solution vector x that is closest to x_* in the 2-norm sense. What is interesting is that when $n < m$ there are infinitely many solutions x_* to the system $Ax = b$, but the minimizer u of (8.4) does not depend on the particular x_* used.

The system (8.1) and methods derived from it are often labeled with NR (N for "Normal" and R for "Residual") while (8.3) and related techniques are labeled with NE (N for "Normal" and E for "Error"). If A is square and nonsingular, the coefficient matrices of these systems are both Symmetric Positive Definite, and the simpler methods for symmetric problems, such as the Conjugate Gradient algorithm, can be applied. Thus, CGNE denotes the Conjugate Gradient method applied to the system (8.3) and CGNR the Conjugate Gradient method applied to (8.1).

There are several alternative ways to formulate symmetric linear systems having the same solution as the original system. For instance, the symmetric linear system

$$\begin{pmatrix} I & A \\ A^T & O \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix} \tag{8.5}$$

with $r = b - Ax$, arises from the standard necessary conditions satisfied by the solution of the constrained optimization problem,

$$\text{minimize} \quad \frac{1}{2}\|r - b\|_2^2 \tag{8.6}$$

$$\text{subject to} \;\; A^T r = 0. \tag{8.7}$$

The solution x to (8.5) is the vector of Lagrange multipliers for the above problem.

Another equivalent symmetric system is of the form

$$\begin{pmatrix} O & A \\ A^T & O \end{pmatrix} \begin{pmatrix} Ax \\ x \end{pmatrix} = \begin{pmatrix} b \\ A^T b \end{pmatrix}.$$

The eigenvalues of the coefficient matrix for this system are $\pm\sigma_i$, where σ_i is an arbitrary singular value of A. Indefinite systems of this sort are not easier to solve than the original nonsymmetric system in general. Although not obvious immediately, this approach is similar in nature to the approach (8.1) and the corresponding Conjugate Gradient iterations applied to them should behave similarly.

A general consensus is that solving the normal equations can be an inefficient approach in the case when A is poorly conditioned. Indeed, the 2-norm condition number of $A^T A$ is given by

$$\mathrm{Cond}_2(A^T A) = \|A^T A\|_2 \, \|(A^T A)^{-1}\|_2.$$

Now observe that $\|A^T A\|_2 = \sigma_{max}^2(A)$ where $\sigma_{max}(A)$ is the largest singular value of A

which, incidentally, is also equal to the 2-norm of A. Thus, using a similar argument for the inverse $(A^T A)^{-1}$ yields

$$\mathrm{Cond}_2(A^T A) = \|A\|_2^2 \, \|A^{-1}\|_2^2 = \mathrm{Cond}_2^2(A). \tag{8.8}$$

The 2-norm condition number for $A^T A$ is exactly the square of the condition number of A, which could cause difficulties. For example, if originally $\mathrm{Cond}_2(A) = 10^8$, then an iterative method may be able to perform reasonably well. However, a condition number of 10^{16} can be much more difficult to handle by a standard iterative method. That is because any progress made in one step of the iterative procedure may be annihilated by the noise due to numerical errors. On the other hand, if the original matrix has a good 2-norm condition number, then the normal equation approach should not cause any serious difficulties. In the extreme case when A is unitary, i.e., when $A^H A = I$, then the normal equations are clearly the best approach (the Conjugate Gradient method will converge in zero step!).

8.2 ROW PROJECTION METHODS

When implementing a basic relaxation scheme, such as Jacobi or SOR, to solve the linear system

$$A^T A x = A^T b, \tag{8.9}$$

or

$$A A^T u = b, \tag{8.10}$$

it is possible to exploit the fact that the matrices $A^T A$ or AA^T need not be formed explicitly. As will be seen, only a row or a column of A at a time is needed at a given relaxation step. These methods are known as *row projection methods* since they are indeed projection methods on rows of A or A^T. Block row projection methods can also be defined similarly.

8.2.1 GAUSS-SEIDEL ON THE NORMAL EQUATIONS

It was stated above that in order to use relaxation schemes on the normal equations, only access to one column of A at a time is needed for (8.9) and one row at a time for (8.10). This is now explained for (8.10) first. Starting from an approximation to the solution of (8.10), a basic relaxation-based iterative procedure modifies its components in a certain order using a succession of relaxation steps of the simple form

$$u_{new} = u + \delta_i e_i \tag{8.11}$$

where e_i is the i-th column of the identity matrix. The scalar δ_i is chosen so that the i-th component of the residual vector for (8.10) becomes zero. Therefore,

$$(b - AA^T(u + \delta_i e_i), e_i) = 0 \tag{8.12}$$

which, setting $r = b - AA^T u$, yields,

$$\delta_i = \frac{(r, e_i)}{\|A^T e_i\|_2^2}. \tag{8.13}$$

Denote by β_i the i-th component of b. Then a basic relaxation step consists of taking

$$\delta_i = \frac{\beta_i - (A^T u, A^T e_i)}{\|A^T e_i\|_2^2}. \tag{8.14}$$

Also, (8.11) can be rewritten in terms of x-variables as follows:

$$x_{new} = x + \delta_i A^T e_i. \tag{8.15}$$

The auxiliary variable u has now been removed from the scene and is replaced by the original variable $x = A^T u$.

Consider the implementation of a forward Gauss-Seidel sweep based on (8.15) and (8.13) for a general sparse matrix. The evaluation of δ_i from (8.13) requires the inner product of the current approximation $x = A^T u$ with $A^T e_i$, the i-th row of A. This inner product is inexpensive to compute because $A^T e_i$ is usually sparse. If an acceleration parameter ω is used, we only need to change δ_i into $\omega\delta_i$. Therefore, a forward SOR sweep would be as follows.

ALGORITHM 8.1: Forward NE-SOR Sweep

1. *Choose an initial* x.
2. *For* $i = 1, 2, \ldots, n$ *Do:*
3. $\quad \delta_i = \omega \, \frac{\beta_i - (A^T e_i, x)}{\|A^T e_i\|_2^2}$
4. $\quad x := x + \delta_i A^T e_i$
5. *EndDo*

Note that $A^T e_i$ is a vector equal to the transpose of the i-th row of A. All that is needed is the row data structure for A to implement the above algorithm. Denoting by nz_i the number of nonzero elements in the i-th row of A, then each step of the above sweep requires $2nz_i + 2$ operations in line 3, and another $2nz_i$ operations in line 4, bringing the total to $4nz_i + 2$. The total for a whole sweep becomes $4nz + 2n$ operations, where nz represents the total number of nonzero elements of A. Twice as many operations are required for the Symmetric Gauss-Seidel or the SSOR iteration. Storage consists of the right-hand side, the vector x, and possibly an additional vector to store the 2-norms of the rows of A. A better alternative would be to rescale each row by its 2-norm at the start.

Similarly, a Gauss-Seidel sweep for (8.9) would consist of a succession of steps of the form

$$x_{new} = x + \delta_i e_i. \tag{8.16}$$

Again, the scalar δ_i is to be selected so that the i-th component of the residual vector for (8.9) becomes zero, which yields

$$(A^T b - A^T A(x + \delta_i e_i), e_i) = 0. \tag{8.17}$$

With $r \equiv b - Ax$, this becomes $(A^T(r - \delta_i A e_i), e_i) = 0$, which yields

$$\delta_i = \frac{(r, Ae_i)}{\|Ae_i\|_2^2}. \tag{8.18}$$

Then the following algorithm is obtained.

ALGORITHM 8.2: Forward NR-SOR Sweep

1. *Choose an initial* x, *compute* $r := b - Ax$.
2. *For* $i = 1, 2, \ldots, n$ *Do:*
3. $\quad \delta_i = \omega \frac{(r, Ae_i)}{\|Ae_i\|_2^2}$
4. $\quad x := x + \delta_i e_i$
5. $\quad r := r - \delta_i A e_i$
6. *EndDo*

In contrast with Algorithm 8.1, the column data structure of A is now needed for the implementation instead of its row data structure. Here, the right-hand side b can be overwritten by the residual vector r, so the storage requirement is essentially the same as in the previous case. In the NE version, the scalar $\beta_i - (x, a_i)$ is just the i-th component of the current residual vector $r = b - Ax$. As a result, stopping criteria can be built for both algorithms based on either the residual vector or the variation in the solution. Note that the matrices AA^T and $A^T A$ can be dense or generally much less sparse than A, yet the cost of the above implementations depends only on the nonzero structure of A. This is a significant advantage of relaxation-type preconditioners over incomplete factorization preconditioners when using Conjugate Gradient methods to solve the normal equations.

One question remains concerning the acceleration of the above relaxation schemes by under- or over-relaxation. If the usual acceleration parameter ω is introduced, then we only have to multiply the scalars δ_i in the previous algorithms by ω. One serious difficulty here is to determine the optimal relaxation factor. If nothing in particular is known about the matrix AA^T, then the method will converge for any ω lying strictly between 0 and 2, as was seen in Chapter 4, because the matrix is positive definite. Moreover, another unanswered question is how convergence can be affected by various reorderings of the rows. For general sparse matrices, the answer is not known.

8.2.2 CIMMINO'S METHOD

In a Jacobi iteration for the system (8.9), the components of the new iterate satisfy the following condition:

$$(A^T b - A^T A(x + \delta_i e_i), e_i) = 0. \tag{8.19}$$

This yields

$$(b - A(x + \delta_i e_i), Ae_i) = 0 \quad \text{or} \quad (r - \delta_i A e_i, Ae_i) = 0$$

in which r is the old residual $b - Ax$. As a result, the i-component of the new iterate x_{new} is given by

$$x_{new,i} = x_i + \delta_i e_i, \tag{8.20}$$

$$\delta_i = \frac{(r, Ae_i)}{\|Ae_i\|_2^2}. \tag{8.21}$$

Here, be aware that these equations do not result in the same approximation as that produced by Algorithm 8.2, even though the modifications are given by the same formula. Indeed, the vector x is not updated after each step and therefore the scalars δ_i are different for the two algorithms. This algorithm is usually described with an acceleration parameter ω, i.e., all δ_i's are multiplied uniformly by a certain ω. If d denotes the vector with coordinates $\delta_i, i = 1, \ldots, n$, the following algorithm results.

ALGORITHM 8.3: Cimmino-NR

1. *Choose initial guess* x_0. *Set* $x = x_0, r = b - Ax_0$
2. *Until convergence Do:*
3.   *For* $i = 1, \ldots, n$ *Do:*
4. $\delta_i = \omega \frac{(r, Ae_i)}{\|Ae_i\|_2^2}$
5.   *EndDo*
6. $x := x + d$ *where* $d = \sum_{i=1}^n \delta_i e_i$
7. $r := r - Ad$
8. *EndDo*

Notice that all the coordinates will use the same residual vector r to compute the updates δ_i. When $\omega = 1$, each instance of the above formulas is mathematically equivalent to performing a projection step for solving $Ax = b$ with $\mathcal{K} = \text{span}\{e_i\}$, and $\mathcal{L} = A\mathcal{K}$. It is also mathematically equivalent to performing an orthogonal projection step for solving $A^T Ax = A^T b$ with $\mathcal{K} = \text{span}\{e_i\}$.

It is interesting to note that when each column Ae_i is normalized by its 2-norm, i.e., if $\|Ae_i\|_2 = 1, i = 1, \ldots, n$, then $\delta_i = \omega(r, Ae_i) = \omega(A^T r, e_i)$. In this situation,

$$d = \omega A^T r = \omega A^T (b - Ax)$$

and the main loop of the algorithm takes the vector form

$$d := \omega A^T r$$
$$x := x + d$$
$$r := r - Ad.$$

Each iteration is therefore equivalent to a step of the form

$$x_{new} = x + \omega \left(A^T b - A^T Ax \right)$$

which is nothing but the Richardson iteration applied to the normal equations (8.1). In particular, as was seen in 4.1, convergence is guaranteed for any ω which satisfies,

$$0 < \omega < \frac{2}{\lambda_{max}} \tag{8.22}$$

where λ_{max} is the largest eigenvalue of $A^T A$. In addition, the best acceleration parameter is given by

$$\omega_{opt} = \frac{2}{\lambda_{min} + \lambda_{max}}$$

in which, similarly, λ_{min} is the smallest eigenvalue of $A^T A$. If the columns are not normalized by their 2-norms, then the procedure is equivalent to a preconditioned Richardson iteration with diagonal preconditioning. The theory regarding convergence is similar but involves the preconditioned matrix or, equivalently, the matrix A' obtained from A by normalizing its columns.

The algorithm can be expressed in terms of projectors. Observe that the new residual satisfies

$$r_{new} = r - \sum_{i=1}^{n} \omega \frac{(r, Ae_i)}{\|Ae_i\|_2^2} \, Ae_i. \tag{8.23}$$

Each of the operators

$$P_i \; : \quad r \longrightarrow \frac{(r, Ae_i)}{\|Ae_i\|_2^2} \, Ae_i \equiv P_i r \tag{8.24}$$

is an orthogonal projector onto Ae_i, the i-th column of A. Hence, we can write

$$r_{new} = \left(I - \omega \sum_{i=1}^{n} P_i \right) r. \tag{8.25}$$

There are two important variations to the above scheme. First, because the point Jacobi iteration can be very slow, it may be preferable to work with sets of vectors instead. Let $\pi_1, \pi_2, \ldots, \pi_p$ be a partition of the set $\{1, 2, \ldots, n\}$ and, for each π_j, let E_j be the matrix obtained by extracting the columns of the identity matrix whose indices belong to π_j. Going back to the projection framework, define $A_i = AE_i$. If an orthogonal projection method is used onto E_j to solve (8.1), then the new iterate is given by

$$x_{new} = x + \omega \sum_{i}^{p} E_i d_i \tag{8.26}$$

$$d_i = (E_i^T A^T A E_i)^{-1} E_i^T A^T r = (A_i^T A_i)^{-1} A_i^T r. \tag{8.27}$$

Each individual block-component d_i can be obtained by solving a least-squares problem

$$\min_d \|r - A_i d\|_2.$$

An interpretation of this indicates that each individual substep attempts to reduce the residual as much as possible by taking linear combinations from specific columns of A_i. Similar to the scalar iteration, we also have

$$r_{new} = \left(I - \omega \sum_{i=1}^{n} P_i \right) r$$

where P_i now represents an orthogonal projector onto the span of A_i.

Note that $A_1, A_2, \ldots, A_p$ is a partition of the column-set $\{Ae_i\}_{i=1,\ldots,n}$ and this partition can be arbitrary. Another remark is that the original Cimmino method was formulated for rows instead of columns, i.e., it was based on (8.1) instead of (8.3). The alternative algorithm based on columns rather than rows is easy to derive.

8.3 CONJUGATE GRADIENT AND NORMAL EQUATIONS

A popular combination to solve nonsymmetric linear systems applies the Conjugate Gradient algorithm to solve either (8.1) or (8.3). As is shown next, the resulting algorithms can be rearranged because of the particular nature of the coefficient matrices.

8.3.1 CGNR

We begin with the Conjugate Gradient algorithm applied to (8.1). Applying CG directly to the system and denoting by z_i the residual vector at step i (instead of r_i) results in the following sequence of operations:

- $\alpha_j := (z_j, z_j)/(A^T A p_j, p_j) = (z_j, z_j)/(A p_j, A p_j)$
- $x_{j+1} := x_j + \alpha_j p_j$
- $z_{j+1} := z_j - \alpha_j A^T A p_j$
- $\beta_j := (z_{j+1}, z_{j+1})/(z_j, z_j)$
- $p_{j+1} := z_{j+1} + \beta_j p_j$.

If the original residual $r_i = b - A x_i$ must be available at every step, we may compute the residual z_{i+1} in two parts: $r_{j+1} := r_j - \alpha_j A p_j$ and then $z_{i+1} = A^T r_{i+1}$ which is the residual for the normal equations (8.1). It is also convenient to introduce the vector $w_i = A p_i$. With these definitions, the algorithm can be cast in the following form.

ALGORITHM 8.4: CGNR

1. *Compute* $r_0 = b - A x_0$, $z_0 = A^T r_0$, $p_0 = z_0$.
2. *For* $i = 0, \ldots,$ *until convergence Do:*
3. $\quad w_i = A p_i$
4. $\quad \alpha_i = \|z_i\|^2 / \|w_i\|_2^2$
5. $\quad x_{i+1} = x_i + \alpha_i p_i$
6. $\quad r_{i+1} = r_i - \alpha_i w_i$
7. $\quad z_{i+1} = A^T r_{i+1}$
8. $\quad \beta_i = \|z_{i+1}\|_2^2 / \|z_i\|_2^2$,
9. $\quad p_{i+1} = z_{i+1} + \beta_i p_i$
10. *EndDo*

In Chapter 6, the approximation x_m produced at the m-th step of the Conjugate Gradient algorithm was shown to minimize the energy norm of the error over an affine Krylov subspace. In this case, x_m minimizes the function

$$f(x) \equiv (A^T A(x_* - x), (x_* - x))$$

over all vectors x in the affine Krylov subspace

$$x_0 + \mathcal{K}_m(A^T A, A^T r_0) = x_0 + \text{span}\{A^T r_0, A^T A A^T r_0, \ldots, (A^T A)^{m-1} A^T r_0\},$$

in which $r_0 = b - Ax_0$ is the initial residual with respect to the original equations $Ax = b$, and $A^T r_0$ is the residual with respect to the normal equations $A^T Ax = A^T b$. However, observe that

$$f(x) = (A(x_* - x), A(x_* - x)) = \|b - Ax\|_2^2.$$

Therefore, CGNR produces the approximate solution in the above subspace which has the smallest residual norm with respect to the original linear system $Ax = b$. The difference with the GMRES algorithm seen in Chapter 6, is the subspace in which the residual norm is minimized.

Example 8.1 Table 8.1 shows the results of applying the CGNR algorithm with no preconditioning to three of the test problems described in Section 3.7.

Matrix	Iters	Kflops	Residual	Error
F2DA	300	4847	0.23E+02	0.62E+00
F3D	300	23704	0.42E+00	0.15E+00
ORS	300	5981	0.30E+02	0.60E-02

Table 8.1 *A test run of CGNR with no preconditioning.*

See Example 6.1 for the meaning of the column headers in the table. The method failed to converge in less than 300 steps for all three problems. Failures of this type, characterized by very slow convergence, are rather common for CGNE and CGNR applied to problems arising from partial differential equations. Preconditioning should improve performance somewhat but, as will be seen in Chapter 10, normal equations are also difficult to precondition.

8.3.2 CGNE

A similar reorganization of the CG algorithm is possible for the system (8.3) as well. Applying the CG algorithm directly to (8.3) and denoting by q_i the conjugate directions, the actual CG iteration for the u variable would be as follows:

- $\alpha_j := (r_j, r_j)/(AA^T q_j, q_j) = (r_j, r_j)/(A^T q_j, A^T q_j)$
- $u_{j+1} := u_j + \alpha_j q_j$
- $r_{j+1} := r_j - \alpha_j AA^T q_j$
- $\beta_j := (r_{j+1}, r_{j+1})/(r_j, r_j)$

- $q_{j+1} := r_{j+1} + \beta_j q_j$.

Notice that an iteration can be written with the original variable $x_i = x_0 + A^T(u_i - u_0)$ by introducing the vector $p_i = A^T q_i$. Then, the residual vectors for the vectors x_i and u_i are the same. No longer are the q_i vectors needed because the p_i's can be obtained as $p_{j+1} := A^T r_{j+1} + \beta_j p_j$. The resulting algorithm described below, the Conjugate Gradient for the normal equations (CGNE), is also known as Craig's method.

ALGORITHM 8.5: CGNE (Craig's Method)

1. *Compute* $r_0 = b - Ax_0$, $p_0 = A^T r_0$.
2. *For* $i = 0, 1, \ldots,$ *until convergence Do:*
3. $\quad \alpha_i = (r_i, r_i)/(p_i, p_i)$
4. $\quad x_{i+1} = x_i + \alpha_i p_i$
5. $\quad r_{i+1} = r_i - \alpha_i A p_i$
6. $\quad \beta_i = (r_{i+1}, r_{i+1})/(r_i, r_i)$
7. $\quad p_{i+1} = A^T r_{i+1} + \beta_i p_i$
8. *EndDo*

We now explore the optimality properties of this algorithm, as was done for CGNR. The approximation u_m related to the variable x_m by $x_m = A^T u_m$ is the actual m-th CG approximation for the linear system (8.3). Therefore, it minimizes the energy norm of the error on the Krylov subspace $\mathcal{K}_m$. In this case, u_m minimizes the function

$$f(u) \equiv (AA^T(u_* - u), (u_* - u))$$

over all vectors u in the affine Krylov subspace,

$$u_0 + \mathcal{K}_m(AA^T, r_0) = u_0 + \text{span}\{r_0, AA^T r_0, \ldots, (AA^T)^{m-1} r_0\}.$$

Notice that $r_0 = b - AA^T u_0 = b - Ax_0$. Also, observe that

$$f(u) = (A^T(u_* - u), A^T(u_* - u)) = \|x_* - x\|_2^2,$$

where $x = A^T u$. Therefore, CGNE produces the approximate solution in the subspace

$$x_0 + A^T \mathcal{K}_m(AA^T, r_0) = x_0 + \mathcal{K}_m(A^T A, A^T r_0)$$

which has the smallest 2-norm of the error. In addition, note that the subspace $x_0 + \mathcal{K}_m(A^T A, A^T r_0)$ is identical with the subspace found for CGNR. Therefore, *the two methods find approximations from the same subspace which achieve different optimality properties: minimal residual for CGNR and minimal error for CGNE.*

8.4 SADDLE-POINT PROBLEMS

Now consider the equivalent system

$$\begin{pmatrix} I & A \\ A^T & O \end{pmatrix} \begin{pmatrix} r \\ x \end{pmatrix} = \begin{pmatrix} b \\ 0 \end{pmatrix}$$

with $r = b - Ax$. This system can be derived from the necessary conditions applied to the constrained least-squares problem (8.6–8.7). Thus, the 2-norm of $b - r = Ax$ is minimized implicitly under the constraint $A^T r = 0$. Note that A does not have to be a square matrix.

This can be extended into a more general constrained quadratic optimization problem as follows:

$$\text{minimize } f(x) \equiv \frac{1}{2}(Ax, x) - (x, b) \tag{8.28}$$

$$\text{subject to } B^T x = c. \tag{8.29}$$

The necessary conditions for optimality yield the linear system

$$\begin{pmatrix} A & B \\ B^T & O \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b \\ c \end{pmatrix} \tag{8.30}$$

in which the names of the variables r, x are changed into x, y for notational convenience. It is assumed that the column dimension of B does not exceed its row dimension. The Lagrangian for the above optimization problem is

$$L(x, y) = \frac{1}{2}(Ax, x) - (x, b) + (y, (B^T x - c))$$

and the solution of (8.30) is the saddle point of the above Lagrangian. Optimization problems of the form (8.28–8.29) and the corresponding linear systems (8.30) are important and arise in many applications. Because they are intimately related to the normal equations, we discuss them briefly here.

In the context of fluid dynamics, a well known iteration technique for solving the linear system (8.30) is Uzawa's method, which resembles a relaxed block SOR iteration.

ALGORITHM 8.6: Uzawa's Method

1. *Choose* x_0, y_0
2. *For* $k = 0, 1, \ldots,$ *until convergence Do:*
3. $\quad x_{k+1} = A^{-1}(b - By_k)$
4. $\quad y_{k+1} = y_k + \omega(B^T x_{k+1} - c)$
5. *EndDo*

The algorithm requires the solution of the linear system

$$Ax_{k+1} = b - By_k \tag{8.31}$$

at each iteration. By substituting the result of line 3 into line 4, the x_k iterates can be eliminated to obtain the following relation for the y_k's,

$$y_{k+1} = y_k + \omega \left(B^T A^{-1}(b - By_k) - c \right)$$

which is nothing but a Richardson iteration for solving the linear system

$$B^T A^{-1} B y = B^T A^{-1} b - c. \tag{8.32}$$

Apart from a sign, this system is the reduced system resulting from eliminating the x variable from (8.30). Convergence results can be derived from the analysis of the Richardson iteration.

COROLLARY 8.1 *Let A be a Symmetric Positive Definite matrix and B a matrix of full rank. Then $S = B^T A^{-1} B$ is also Symmetric Positive Definite and Uzawa's algorithm converges, if and only if*

$$0 < \omega < \frac{2}{\lambda_{max}(S)}. \tag{8.33}$$

In addition, the optimal convergence parameter ω is given by

$$\omega_{opt} = \frac{2}{\lambda_{min}(S) + \lambda_{max}(S)}.$$

Proof. The proof of this result is straightforward and is based on the results seen in Example 4.1. ∎

It is interesting to observe that when $c = 0$ and A is Symmetric Positive Definite, then the system (8.32) can be regarded as the normal equations for minimizing the A^{-1}-norm of $b - By$. Indeed, the optimality conditions are equivalent to the orthogonality conditions

$$(b - By, Bw)_{A^{-1}} = 0, \quad \forall\, w,$$

which translate into the linear system $B^T A^{-1} B y = B^T A^{-1} b$. As a consequence, the problem will tend to be easier to solve if the columns of B are almost orthogonal with respect to the A^{-1} inner product. This is true when solving the *Stokes problem* where B represents the discretization of the gradient operator while B^T discretizes the divergence operator, and A is the discretization of a Laplacian. In this case, if it were not for the boundary conditions, the matrix $B^T A^{-1} B$ would be the identity. This feature can be exploited in developing preconditioners for solving problems of the form (8.30). Another particular case is when A is the identity matrix and $c = 0$. Then, the linear system (8.32) becomes the system of the normal equations for minimizing the 2-norm of $b - By$. These relations provide insight in understanding that the block form (8.30) is actually a form of normal equations for solving $By = b$ in the least-squares sense. However, a different inner product is used.

In Uzawa's method, a linear system at each step must be solved, namely, the system (8.31). Solving this system is equivalent to finding the minimum of the quadratic function

$$\text{minimize} \quad f_k(x) \;\equiv\; \frac{1}{2}(Ax, x) - (x, b - By_k). \tag{8.34}$$

Apart from constants, $f_k(x)$ is the Lagrangian evaluated at the previous y iterate. The solution of (8.31), or the equivalent optimization problem (8.34), is expensive. A common alternative replaces the x-variable update (8.31) by taking one step in the gradient direction for the quadratic function (8.34), usually with fixed step-length ϵ. The gradient of $f_k(x)$ at the current iterate is $Ax_k - (b - By_k)$. This results in the Arrow-Hurwicz Algorithm.

ALGORITHM 8.7: The Arrow-Hurwicz algorithm

1. *Select an initial guess x_0, y_0 to the system (8.30)*
2. *For $k = 0, 1, \ldots,$ until convergence Do:*
3. *Compute $x_{k+1} = x_k + \epsilon(b - Ax_k - By_k)$*
4. *Compute $y_{k+1} = y_k + \omega(B^T x_{k+1} - c)$*
5. *EndDo*

The above algorithm is a block-iteration of the form

$$\begin{pmatrix} I & O \\ -\omega B^T & I \end{pmatrix} \begin{pmatrix} x_{k+1} \\ y_{k+1} \end{pmatrix} = \begin{pmatrix} I - \epsilon A & -\epsilon B \\ O & I \end{pmatrix} \begin{pmatrix} x_k \\ y_k \end{pmatrix} + \begin{pmatrix} \epsilon b \\ -\omega c \end{pmatrix}.$$

Uzawa's method, and many similar techniques for solving (8.30), are based on solving the reduced system (8.32). An important observation here is that the Schur complement matrix $S \equiv B^T A^{-1} B$ need not be formed explicitly. This can be useful if this reduced system is to be solved by an iterative method. The matrix A is typically factored by a Cholesky-type factorization. The linear systems with the coefficient matrix A can also be solved by a preconditioned Conjugate Gradient method. Of course these systems must then be solved accurately.

Sometimes it is useful to "regularize" the least-squares problem (8.28) by solving the following problem in its place:

$$\text{minimize } f(x) \equiv \frac{1}{2}(Ax, x) - (x, b) + \rho(Cy, y)$$
$$\text{subject to } B^T x = c$$

in which ρ is a scalar parameter. For example, C can be the identity matrix or the matrix $B^T B$. The matrix resulting from the Lagrange multipliers approach then becomes

$$\begin{pmatrix} A & B \\ B^T & \rho C \end{pmatrix}.$$

The new Schur complement matrix is

$$S = \rho C - B^T A^{-1} B.$$

Example 8.2 In the case where $C = B^T B$, the above matrix takes the form

$$S = B^T(\rho I - A^{-1})B.$$

Assuming that A is SPD, S is also positive definite when

$$\rho \geq \frac{1}{\lambda_{min}(A)}.$$

However, it is also *negative definite* for

$$\rho \leq \frac{1}{\lambda_{max}}(A),$$

a condition which may be easier to satisfy on practice.

EXERCISES

1. Derive the linear system (8.5) by expressing the standard necessary conditions for the problem (8.6–8.7).

2. It was stated in Section 8.2.2 that when $\|A^T e_i\|_2 = 1$ for $i = 1, \dots, n$, the vector d defined in Algorithm 8.3 is equal to $\omega A^T r$.

a. What does this become in the general situation when $\|A^T e_i\|_2 \neq 1$?

b. Is Cimmino's method still equivalent to a Richardson iteration?

c. Show convergence results similar to those of the scaled case.

3. In Section 8.2.2, Cimmino's algorithm was derived based on the Normal Residual formulation, i.e., on (8.1). Derive an "NE" formulation, i.e., an algorithm based on Jacobi's method for (8.3).

4. What are the eigenvalues of the matrix (8.5)? Derive a system whose coefficient matrix has the form

$$B(\alpha) = \begin{pmatrix} 2\alpha I & A \\ A^T & O \end{pmatrix}.$$

and which is also equivalent to the original system $Ax = b$. What are the eigenvalues of $B(\alpha)$? Plot the spectral norm of $B(\alpha)$ as a function of α.

5. It was argued in Section 8.4 that when $c = 0$ the system (8.32) is nothing but the normal equations for minimizing the A^{-1}-norm of the residual $r = b - By$.

a. Write the associated CGNR approach for solving this problem. Find a variant that requires only one linear system solution with the matrix A at each CG step [Hint: Write the CG algorithm for the associated normal equations and see how the resulting procedure can be reorganized to save operations]. Find also a variant that is suitable for the case where the Cholesky factorization of A is available.

b. Derive a method for solving the equivalent system (8.30) for the case when $c = 0$ and then for the general case wjen $c \neq 0$. How does this technique compare with Uzawa's method?

6. Consider the linear system (8.30) in which $c = 0$ and B is of full rank. Define the matrix

$$P = I - B(B^T B)^{-1} B^T.$$

a. Show that P is a projector. Is it an orthogonal projector? What are the range and null spaces of P?

b. Show that the unknown x can be found by solving the linear system

$$PAPx = Pb, \tag{8.35}$$

in which the coefficient matrix is singular but the system is consistent, i.e., there is a nontrivial solution because the right-hand side is in the range of the matrix (see Chapter 1).

c. What must be done toadapt the Conjugate Gradient Algorithm for solving the above linear system (which is symmetric, but not positive definite)? In which subspace are the iterates generated from the CG algorithm applied to (8.35)?

d. Assume that the QR factorization of the matrix B is computed. Write an algorithm based on the approach of the previous questions for solving the linear system (8.30).

7. Show that Uzawa's iteration can be formulated as a fixed-point iteration associated with the splitting $C = M - N$ with

$$M = \begin{pmatrix} A & O \\ -\omega B^T & I \end{pmatrix}, \quad N = \begin{pmatrix} O & -B \\ O & I \end{pmatrix}.$$

Derive the convergence result of Corollary 8.1 .

8. Show that each new vector iterate in Cimmino's method is such that

$$x_{new} = x + \omega A^{-1} \sum_i P_i r,$$

where P_i is defined by (8.24).

9. In Uzawa's method a linear system with the matrix A must be solved at each step. Assume that these systems are solved inaccurately by an iterative process. For each linear system the iterative process is applied until the norm of the residual $r_{k+1} = (b - By_k) - Ax_{k+1}$ is less than a certain threshold ϵ_{k+1}.

a. Assume that ω is chosen so that (8.33) is satisfied and that ϵ_k converges to zero as k tends to infinity. Show that the resulting algorithm converges to the solution.

b. Give an explicit upper bound of the error on y_k in the case when ϵ_i is chosen of the form $\epsilon = \alpha^i$, where $\alpha < 1$.

10. Assume $\|b - Ax\|_2$ is to be minimized, in which A is $n \times m$ with $n > m$. Let x_* be the minimizer and $r = b - Ax_*$. What is the minimizer of $\|(b + \alpha r) - Ax\|_2$, where α is an arbitrary scalar?

NOTES AND REFERENCES. Methods based on the normal equations have been among the first to be used for solving nonsymmetric linear systems [130, 58] by iterative methods. The work by Bjork and Elfing [27], and Sameh et al. [131, 37, 36] revived these techniques by showing that they have some advantages from the implementation point of view, and that they can offer good performance for a broad class of problems. In addition, they are also attractive for parallel computers. In [174], a few preconditioning ideas for normal equations were described and these will be covered in Chapter 10. It would be helpful to be able to determine whether or not it is preferable to use the normal equations approach rather than the "direct equations" for a given system, but this may require an eigenvalue/singular value analysis.

It is sometimes argued that the normal equations approach is *always* better, because it has a robust quality which outweighs the additional cost due to the slowness of the method in the generic elliptic case. Unfortunately, this is not true. Although variants of the Kaczmarz and Cimmino algorithms deserve a place in any robust iterative solution package, they cannot be viewed as a panacea. In *most* realistic examples arising from Partial Differential Equations, the normal equations route gives rise to much slower convergence than the Krylov subspace approach for the direct equations. For ill-conditioned problems, these methods will simply fail to converge, unless a good preconditioner is available. ■